MAKE | BUILD | HACK | CREATE

HackSpace

TECHNOLOGY IN YOUR HANDS

hsmag.CC | July 2021 | Issue #44

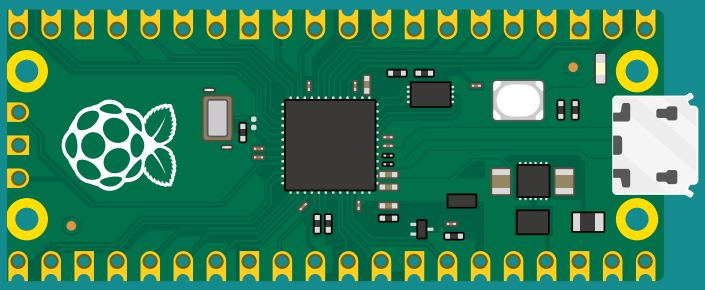
PALLETS

Creative uses for budget wood

PICO AUDIO

Make music on a microcontroller

ARDUINO + PICO



Use your Arduino skills on Raspberry Pi Pico to power EVERYTHING

July 2021
Issue #44 £6

3D PRINTED WIND FARM

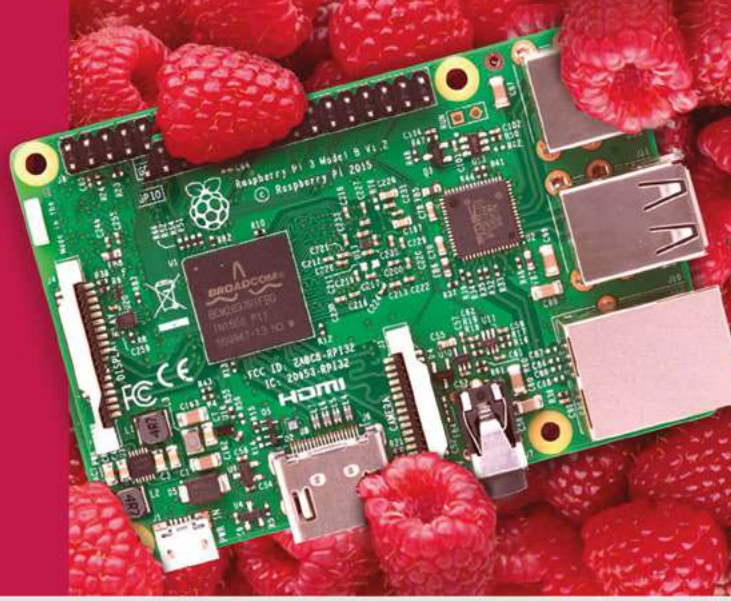
Generate your own electricity

Simone GIERTZ

All hail the queen of robots

CNC MITRE SAWS BLUETOOTH LASERS

American **Raspberry Pi** Shop



- Displays
- HATs
- Sensors
- Cases
- Arcade
- Swag
- Project Kits
- Cameras
- Power Options
- Add-on Boards
- Cables and Connectors
- GPIO and Prototyping

Partner and official reseller for top Pi brands:

adafruit · sparkfun ELECTRONICS · PIMORONI · HiFiBerry · and many others!

Price, service, design, and logistics support for **VOLUME PROJECTS**

USA PiShop.us

Canada BuyaPi.ca

Raspberry Pi APPROVED RESELLER

Welcome to HackSpace magazine

What is the best programming language? That's a great question to ask if you ever want to start an argument. The truth is that the majority of applications can be done equally well in a wide range of languages. Some might make one stage of it a little faster, some might run a little quicker once they're created, some might be a little easier to debug once they're written, but in most cases, you can just pick a language and not worry about it. In truth, the best language is usually the one you already know.

For many hobbyist microcontroller programmers, the language they already know is Arduino (I'll leave it to the linguists to argue about whether or not it's a language, a framework, or a dialect of C++).

In this issue, we take a look at how to program Raspberry Pi Pico using Arduino. You can get set up really quickly, and there are heaps of libraries and example code to get you started, or help you take your sketches to the next level.

BEN EVERARD
Editor ben.everard@raspberrypi.com

Got a comment, question, or thought about HackSpace magazine?

get in touch at **hsmag.cc/hello**

GET IN TOUCH

hackspace@raspberrypi.com
hackspacemag
hackspacemag

ONLINE

hsmag.cc

Available on the App Store

GET IT ON Google Play

EDITORIAL

Editor
Ben Everard
ben.everard@raspberrypi.com

Features Editor
Andrew Gregory
andrew.gregory@raspberrypi.com

Sub-Editors
David Higgs, Nicola King

DESIGN

Critical Media
criticalmedia.co.uk

Head of Design
Lee Allen

Designers
Sam Ribbits, Lucy Cowan, Ty Logan

Photography
Brian O'Halloran

CONTRIBUTORS

Rosie Hattersley, Jo Hinchliffe, Marc de Vinck, Andrew Lewis, Alex Bate, Fabian Steppat, Rob Miles

PUBLISHING

Publishing Director
Russell Barnes
russell@raspberrypi.com

Advertising
Charlie Milligan
charlotte.milligan@raspberrypi.com

DISTRIBUTION

Seymour Distribution Ltd
2 East Poultry Ave,
London EC1A 9PT
+44 (0)207 429 4000

SUBSCRIPTIONS

Unit 6, The Enterprise Centre,
Kelvin Lane, Manor Royal,
Crawley, West Sussex, RH10 9PE

To subscribe
01293 312189
hsmag.cc/subscribe

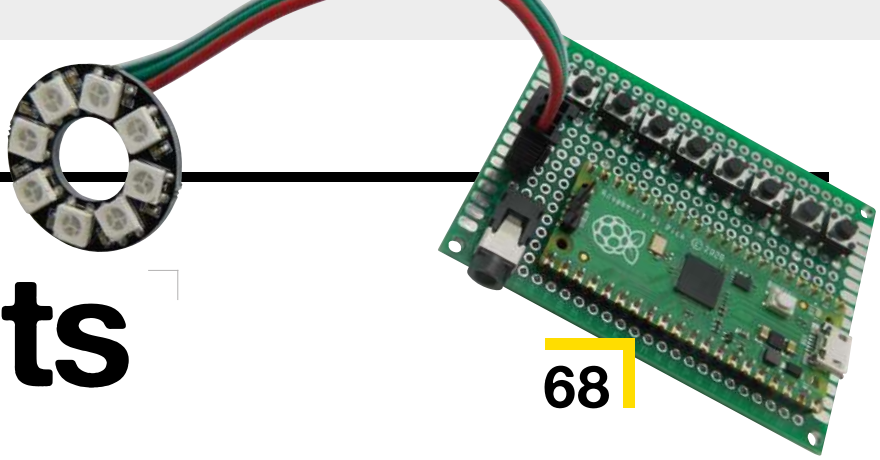
Subscription queries
hackspace@subscriptionhelpline.co.uk

recycle

CC BY NC SA

This magazine is printed on paper sourced from sustainable forests. The printer operates an environmental management system which has been assessed as conforming to ISO 14001.

HackSpace magazine is published by Raspberry Pi (Trading) Ltd., Maurice Wilkes Building, St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products or services referred to or advertised. Except where otherwise noted, content in this magazine is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0). ISSN: 2515-5148.



Contents

06 SPARK

06 **Top Projects**
Beautiful, useful things

18 **Objet 3d'art**
The engineering brilliance of the Wankel engine

20 **Meet the Maker:** Shawn Hymel
The machines are taking over. And he's helping them!

27 **Columns**
Peering into the past

28 **Letters**
Let us know what you think of this here magazine

30 **Kickstarting**
Inky goodness in a 6-inch, WiFi-enabled package

33 LENS

34 **Arduino + Pico**
Add embedded genius to absolutely anything

48 **How I Made:** Windmill & generator
Harness free electrons from the sky with 3D printing

54 **Interview:** Simone Giertz
How useful things are the same (kind of) as silly things

64 **Improviser's Toolbox:** Pallets
Creative uses for scrap wood

68 **In the workshop:** Pico audio
Squeeze sounds out of your tiny microcontroller

Cover Feature

ARDUINO + PICO

The number one embedded system has a new best buddy: the Raspberry Pi Pico

34

Tutorial

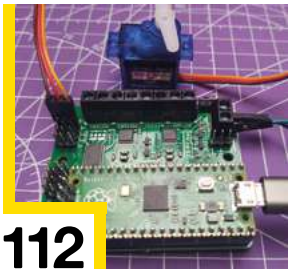
Laser cutting

94 Add an adjustable-height bed for your laser cutter

30

20

06



112

Direct from Shenzhen

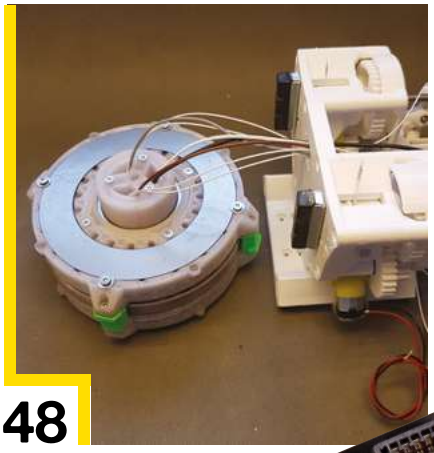
Bluetooth amp

110 A ridiculously affordable way to add tunes to your latest build

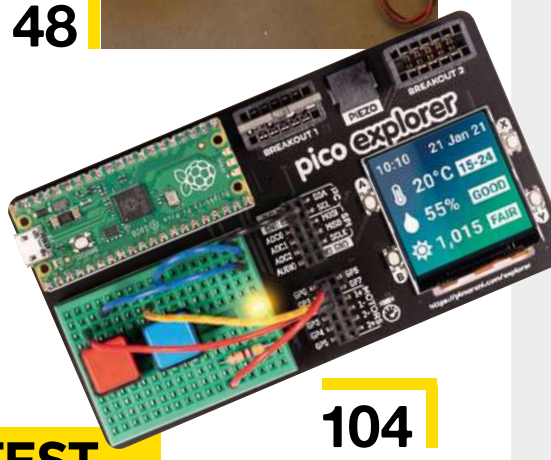
Interview

Simone Giertz

54 How to win at crowdfunding, robotics, and more



48



104

73 FORGE

74 **SoM** Blinka
Make CircuitPython even easier to learn

76 **Tutorial** CNC
Engraving and cutting aluminium

82 **Tutorial** Build an arcade cabinet
Wire up the controls to the Raspberry Pi's GPIO

88 **Tutorial** FreeCAD
Harness the power of arrays

94 **Tutorial** Laser cutter
Build an adjustable-height bed

98 **Tutorial** Hardware control
Send signals to USB via a browser

103 FIELD TEST

104 **Best of Breed** Pico add-ons
Dip into the RP2040 ecosystem

110 **Direct from Shenzhen**
You've heard of *Pet Sounds*: here's cheap sounds

112 **Review** Kitronik Robotics Board for Pico
All you need to build a Terminator-style hellscape

Canine skull mask

By **Willow Creative** hsmag.cc/SkullMask

We came across this while talking to Willow Creative, who we'll be featuring in next issue's Meet the Maker, but it's too creepy to keep to ourselves. It's a 3D-printed dog's skull (a feline version is also available). Merel, the maker, told us that since she started making them, she's sold over 500 of these skulls – that's a Hieronymus Bosch nightmare vision if ever we thought of one.

Right
It may only be June, but it's never too early to start thinking about Halloween

Modern cardboard chair

By Morley Kert

hsmag.cc/MorleyKert

When we were kids, we'd beg and plead for cardboard to play with. We didn't know it, b ut we were practising for 2021, when home deliveries would swamp us with the stuff. One maker who's adapted to modern times is Morley Kert, who's made this chair from 30 cardboard boxes he had spare after moving apartment. The bill of materials for this build is simple – just find 30 cardboard boxes, a sharp knife, permanent marker, and glue, and you too could be reclining in style.

Right
Cardboard is easy to work with compared with metal or wood – just make sure you have spare blades on hand for your Stanley knife

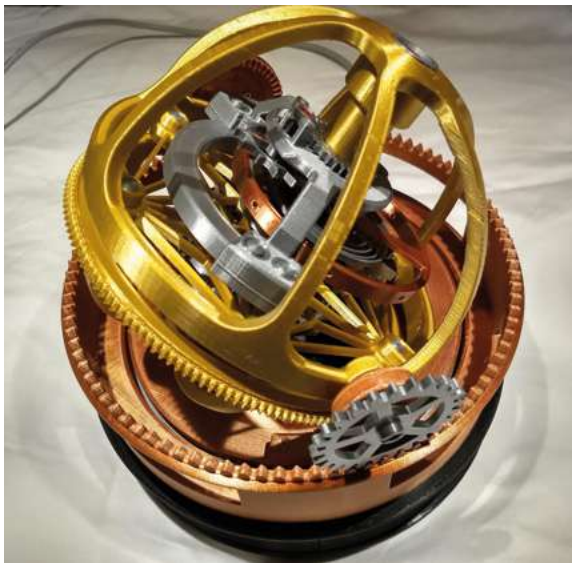
Furious Tourbillon

By McMaven

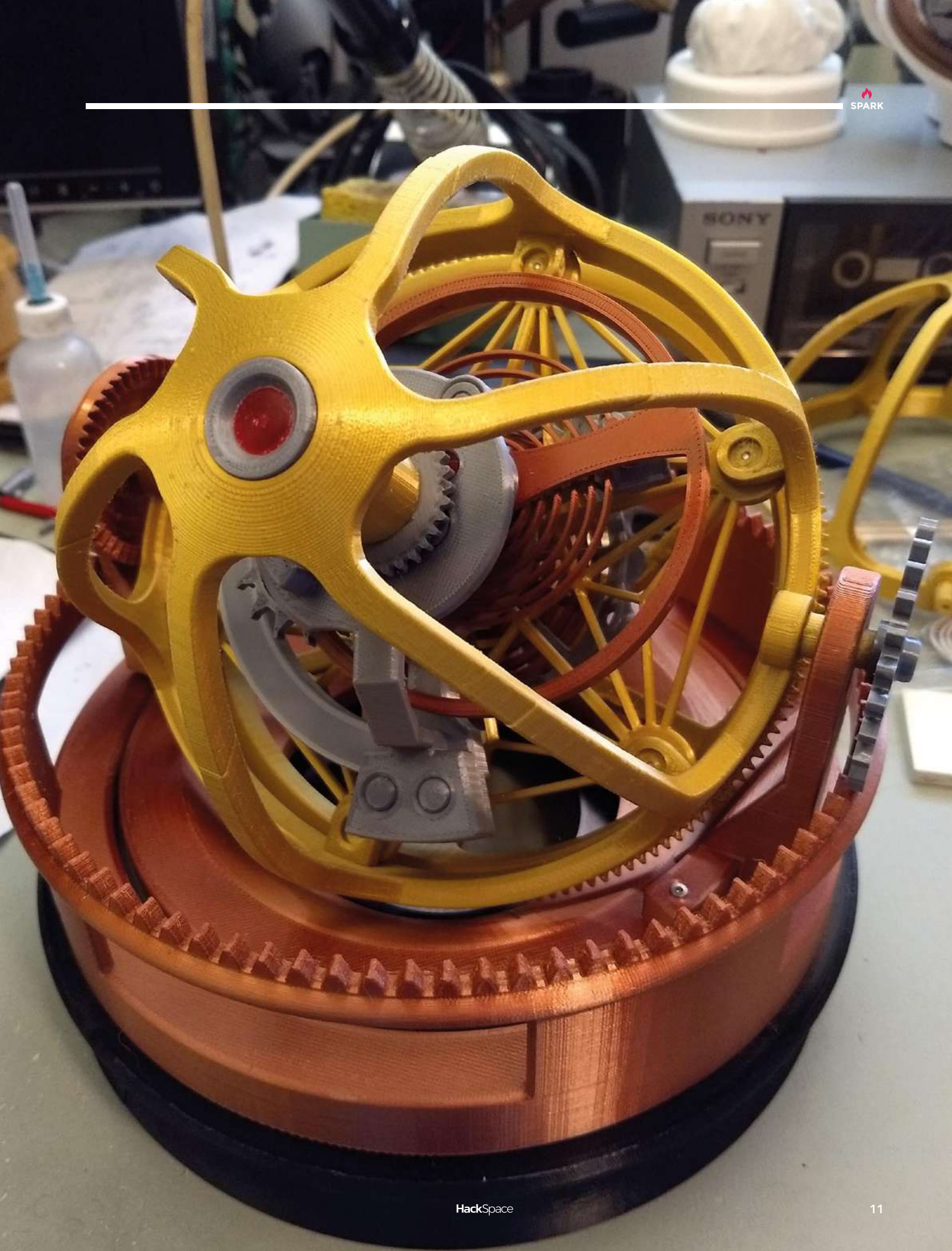
hsmag.cc/Tourbillon

We're not big on luxury timepieces here at HackSpace Mansions: give us a Casio or an RTC module if we need to know the time. However, everyone's different: some people like to spend a fortune on Swiss-made watches full of beautifully made precision parts. If you're one of these (or you'd like to be), here's a fascinating tribute to the Furious Tourbillon watch by Jacob & Co., watchmakers of Geneva.

The model is large – 20 cm in diameter – and printed in luxury silk PLA. We're not sure what it does, but we like it a lot. □



Right
The mechanism is incredibly precise. You could almost say it's like clockwork


SONY

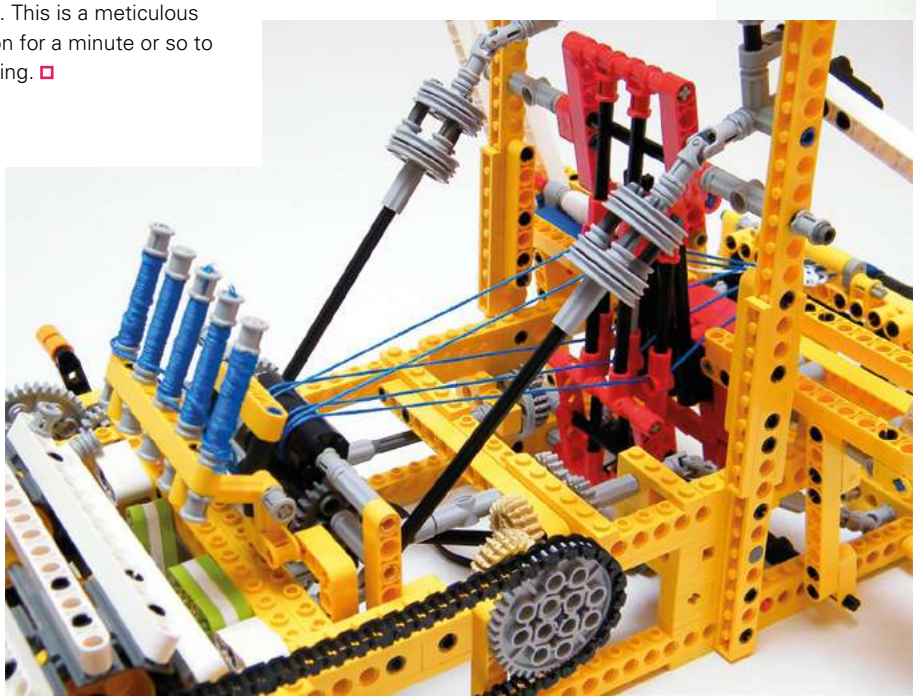
Lego loom

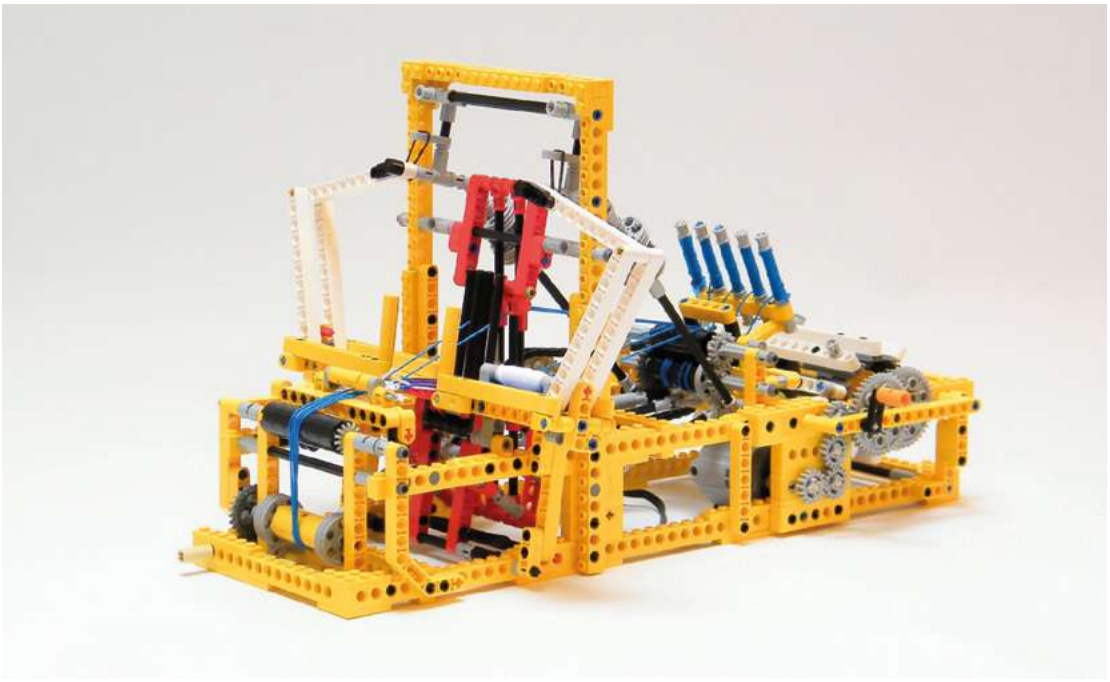
By Nico71

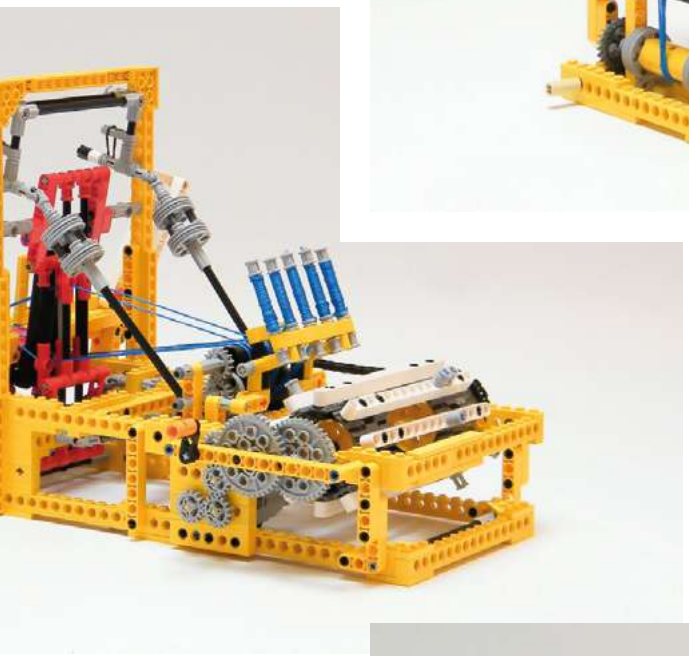
hsmag.cc/LegoLoom

The links between programming and weaving are well known. The punch cards used by Jacquard looms were the first storage medium used by the earliest mechanical computers, so it feels like we've come full circle when we look at this miniature loom by Nico71.

The whole thing is made out of Lego (Technic Lego, to be precise), and it actually works. The cloth it produces is small, but it's woven in exactly the same way as the weavers of the early Industrial Revolution would have recognised. This is a meticulous build, and you really need to watch it in action for a minute or so to understand what's happening: it's mesmerising.







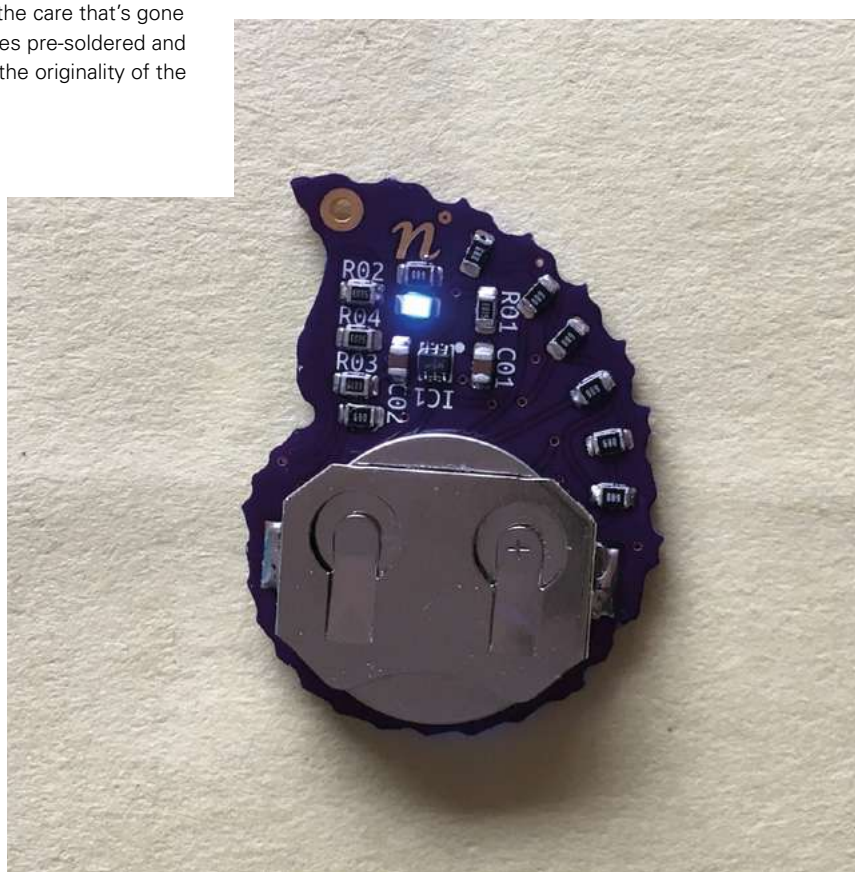
Below
Weave a scarf for a very small friend

Nautilus

By n°Garage

hsmag.cc/Nautilus

This pre-soldered badge is the latest shiny thing to catch our eye on Tindie. It features eight blue LEDs… and not a great deal else. That's not why we like it though: the main reason is the care that's gone into the shape of the PCB. It comes pre-soldered and ready-to-wear, and goes to show the originality of the work that's out there in wearable tech.



Right
Nautilus shells are an example in nature of the golden spiral, an innate beauty that we only glimpse through the Fibonacci sequence

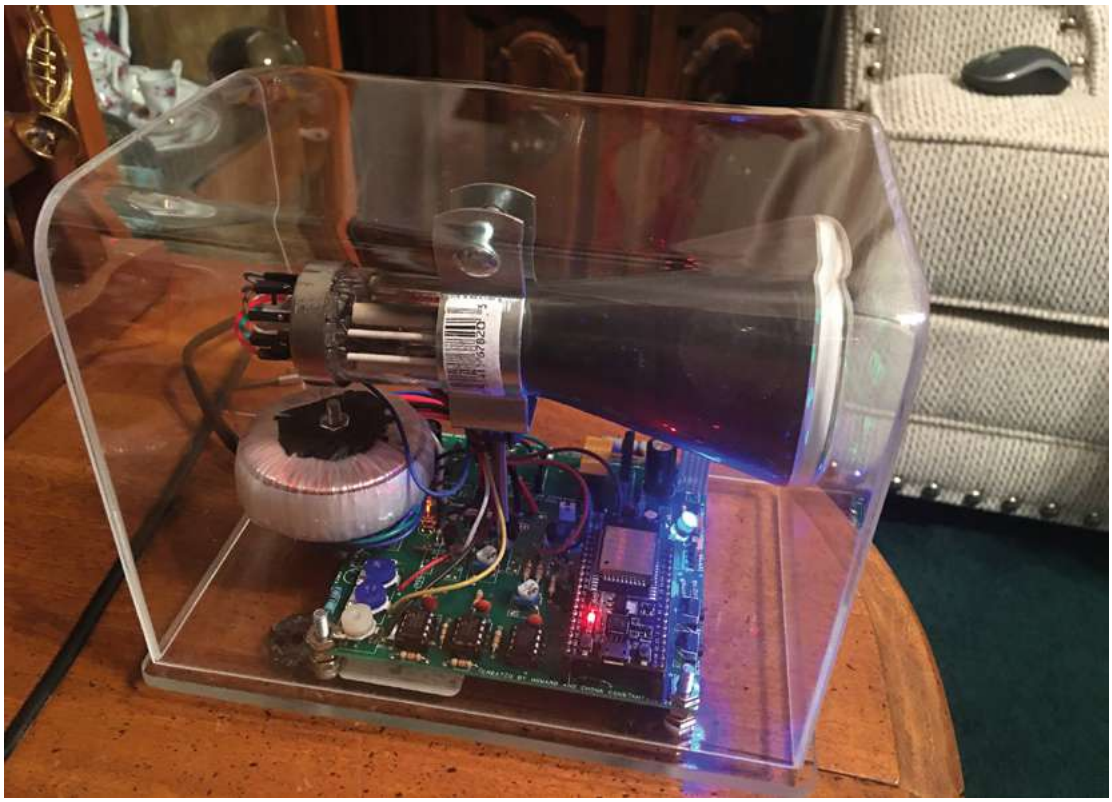
REGULAR

Mini Oscilloscope Clock DG7-6 Cathode Ray Tube 3”

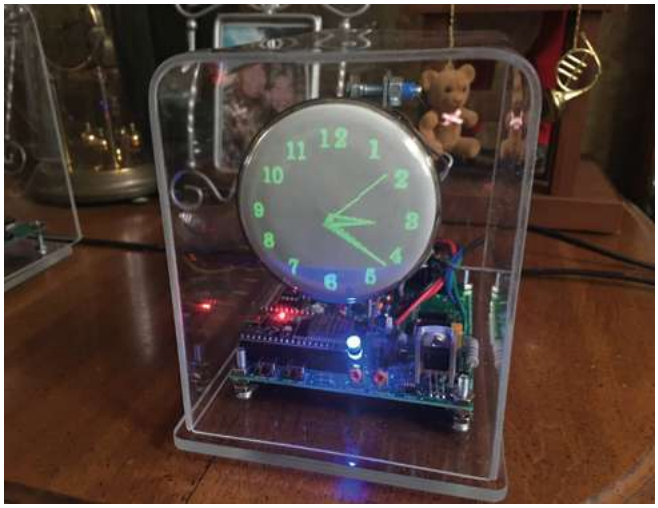
By Oscilloscope clocks

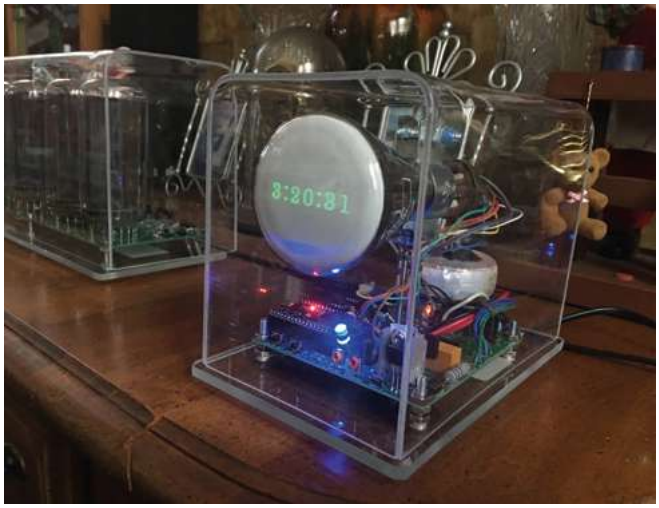
hsmag.cc/OscilloscopeClock

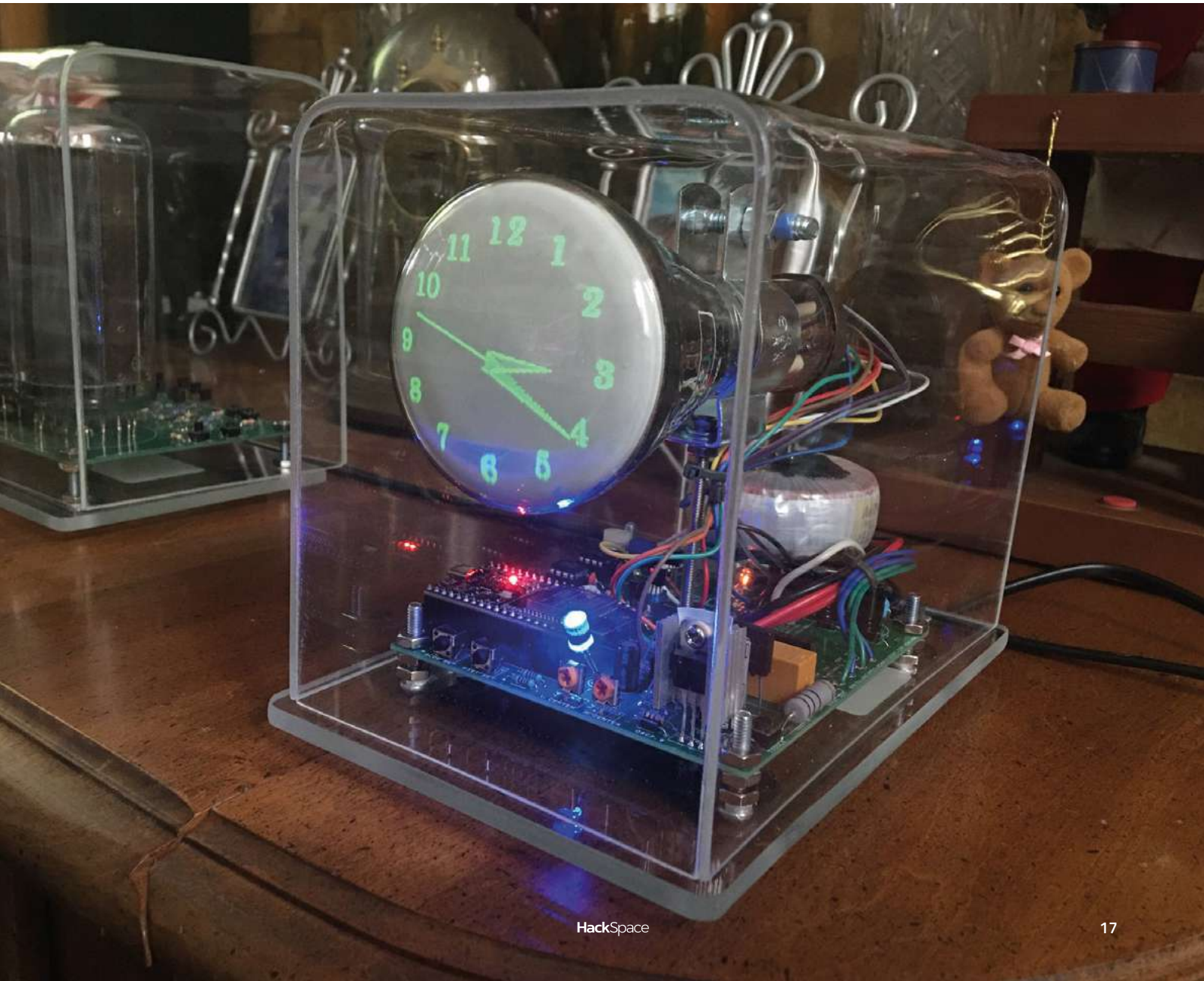
CRTs – **cathode-ray tubes – are a ridiculous, obsolete technology.** They run at a much higher voltage than LEDs – they’re bulky, fragile, and considering their ubiquity in the 1980s, there are relatively few of them around. Still, there’s something evocative about them, so we love this clock. It uses a three-inch CRT display salvaged from an old oscilloscope, brought to new life with an ESP32.



Right
The display moves slightly every six seconds to help prevent ‘burn in’ of the CRT




8:20:81



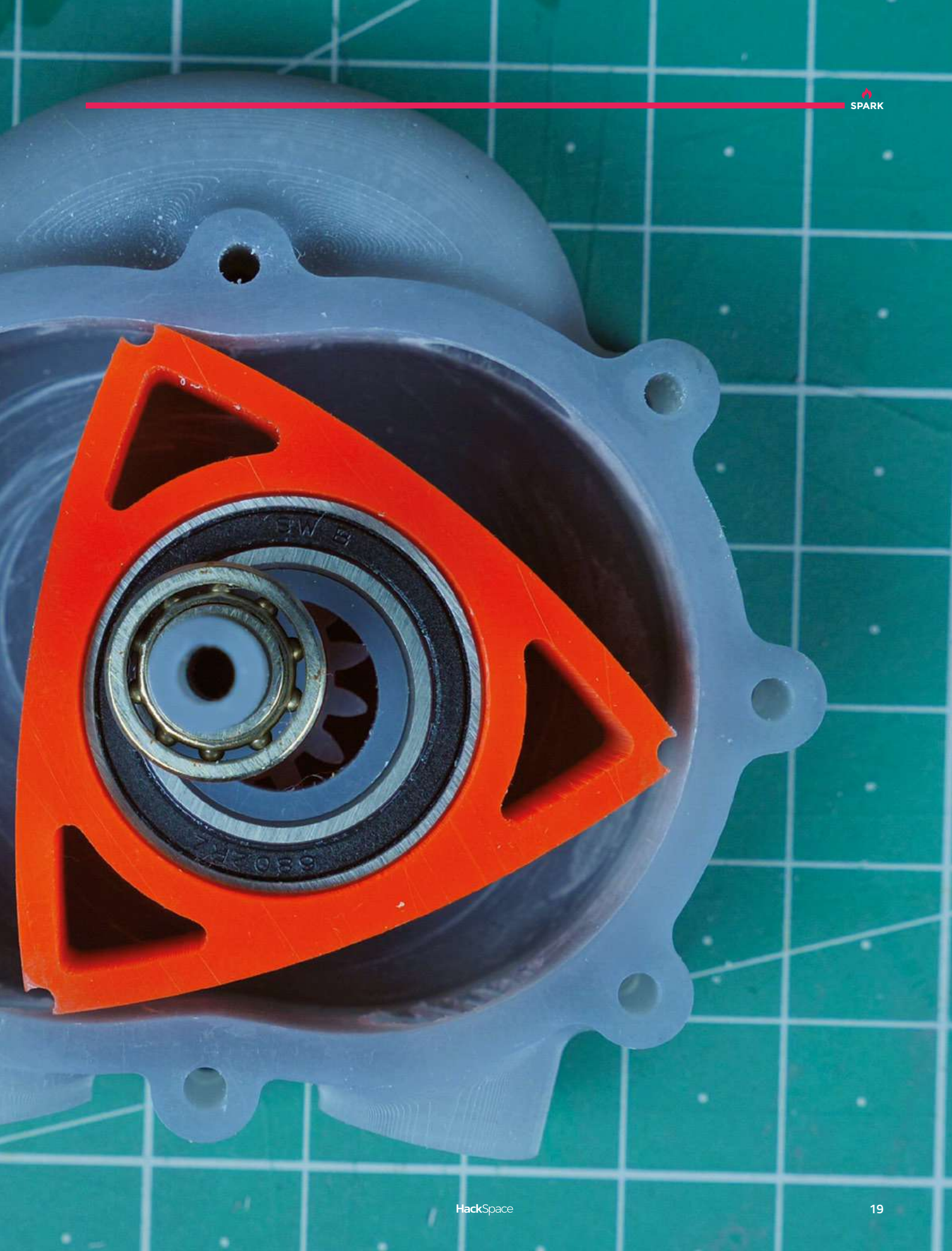
Objet 3d'art

3D-printed artwork to bring more beauty into your life

Instead of pistons moving up and down, with a crank to convert linear motion into rotary motion, a Wankel engine uses a triangular rotor that turns within a chamber. As the triangle rotates, different areas within the chamber get compressed. When petrol and air are injected, and the mixture ignited, the triangle spins fast enough to power a car, motorbike, or aircraft. Wankel engines have advantages (high power-to-weight ratio, low vibration) and disadvantages (poor fuel efficiency, a tendency to break down if not maintained) compared with standard piston engines, but we like them mainly because of the clever, original design – which also translates well into 3D printing.

Joel Gomes has created this 3D-printed model of a Wankel engine, which fits together well enough that it functions as an engine – though instead of using exploding hydrocarbons, he's elected to use compressed CO_2. Joel had to adapt a quick-release valve, tapping a thread to add an electrically controlled release mechanism to enable him to release the gas into the Wankel engine remotely. It's currently powering a remote-controlled car, in a brilliant mix-up of 3D printing and steampunk. □

hsmag.cc/Wankel



REGULAR

Meet The Maker: Shawn Hymel

Educator, electronics engineer, machine learning enthusiast, and a guy who wants you to succeed

Right
Shawn's trained his SNES controller to respond to the verbal command "Hadouken!" – it's a brilliantly silly way to illustrate the power of machine learning

We spoke to Shawn Hymel expecting to hear about the projects he's done. Instead, we got a stream of enthusiasm that made us curious, interested, and excited about the future of machine learning. Anyone can program a smart device and teach it to do whatever they want! That's the impression he left us with, and we're still buzzing over the potential of embedded ML. Now, we just have to get one or two things out of the way before we can use this still-new technology (as Shawn says, it's still just ten years old) to take over the world. Or at least, build a farting pumpkin.

"I loved math and science through school, so going into engineering was a natural fit for me. I really fell in love with the idea of doing computer engineering. From there I got a job doing circuit design, writing software, writing firmware, eventually got my master's in computer engineering and moved to SparkFun, which took all of those ideas to the maker movement – supplying all these electronics to engineers as well as makers, hackers, tinkerers and absolutely loving the idea of helping them get started faster, which is where I fell in love with the idea of teaching all of this stuff.

"The first year and a half I was at SparkFun, I was in the engineering department. I really enjoyed it. We made products, I designed boards, we used EAGLE, and pretty much did everything in Arduino when it came to firmware. We were writing Arduino libraries because the whole idea was to create boards that enable people to make electronics as easily as possible.

"There was something in me that said I wanted to do more videos, I wanted to overcome this fear I had of cameras. An opportunity arose, and I got the chance to fulfil that goal of teaching people electronics – to be the Alton Brown of electronics. If you're into cooking, he's absolutely one of my favourite TV chefs. He goes into a lot of the science of cooking.

"I'm trying to impart the skills to someone who just got out of college, or is trying to continue their education where a college education failed to provide them those skills. For example, I remember several times at college we'd go into a lab and they'd just hand us a soldering iron – 'You have to make this antenna and solder this capacitor between it.' No instructions – just figure it out. How do you use an oscilloscope? Just figure it out. How do you lay out a PCB? They don't teach you that. They do now, but they didn't at the time.

MACHINE LEARNING

"I thoroughly love the Raspberry Pi Pico. In fact, I've got one sitting right here on my desk.

"It's got two cores, so I can do things like machine learning on it, which has been my current kick. You can have one core running your machine learning stuff, and have another cover taking in sound data, sending it off to the other core… you're missing some of that DSP stuff, so it's not as efficient as a Cortex-M4 which has a floating-point unit built in.

"The PIO is genius; I've tinkered with that for a little. I've just released a video talking about using PIO from the C SDK. Essentially, you can take any peripheral up to a certain limit. I'm probably not going to run →


HI
112100
KO
99
RYU
KEN
SPH0645
I2S Mic
3V
GND
BCLK
DOUT
LRCL
SEL
X
A
Y
B
SELECT
START

REGULAR

> Whoever designed that, they get **gold stars across the board because that is such a genius move**

USB 3.0 super speeds on it or HDMI, but I can do things like take a NeoPixel strip, hook it up, and I don't have to spend any cycles on the main core sending out data to this NeoPixel strip any more; I can just have the PIO do it. I just shove information into a FIFO and the PIO just handles it. Whoever designed that, they get gold stars across the board because that is such a genius move.

A LEG UP

"I had a slight leg up into machine learning, though I didn't realise when I did my master's. My master's thesis was in applying hidden Markov models as a way of classifying wireless signals, and taking the hidden Markov model, all the algorithms associated with it, and putting it on a GPU. This was in 2010, and CUDA was only a year old. The ability to write generic code for an NVIDIA graphics card was fairly new at the time. There was another language that let you do it but NVIDIA was like, 'We'll support this and let researchers do what they want with graphics cards.' So I was tinkering with that back in 2010. I had no idea that was machine learning! The lab was full of people tinkering with these Markov models, to try to predict behaviours from observed data sets.

"Years and years later, I realised: that was machine learning. I created a model that would train itself to classify these signals. I technically did my master's on machine learning; I just didn't know it. So I had a bit of a leg up. I took Andrew Ng's course on Coursera (If you ever want to look at the nuts and bolts of how neural networks work, that is the absolute best class, and I can't recommend it more highly).

"That's where it all came to make sense to me. Neural networks are not the same as Markov models, but the idea was the same, so it made sense to me.

"That was 2019. I knew machine learning was coming down the pipe, I knew it was big on the server world. Amazon was using it, Netflix was using it, all these things were out at that time. And we knew they were using machine learning. Obviously they were using machine learning to process what we were saying. But this idea of running it, not on the back-end but on the edge, where I can take it and put it on a microcontroller.

"With the Amazon Alexa, you say the magic word and it pops straight to life. It's not always listening, despite what people think: it's listening locally, looking for one key word. Then, when it hears that key word, it wakes up and starts streaming stuff to a server. The device itself only listens to about a second of data. If it's on memory maybe you can get it, but it's not trying to store everything you're saying up to the point where you say the magic wake word.

"So that magic wake word is running machine learning on a microcontroller, and there are potentially tons of uses for this outside of the normal 'let's →

Right
Shawn gets to play with the tech of the future, today. And so can we!

capture data and send it to a server to process somewhere else', especially if you don't have an internet connection or you want to save bandwidth.

"For example, let's imagine having a camera and I want to detect if there's a person. I don't have to store all that data – can you imagine sending a constant video stream to your network?

"Rather than streaming a bunch of data, let's have each camera be smart enough that they can say 'there's a person in this frame'. If we push this classification down to the edge devices

"So now IoT is a lot smarter as a result, and suddenly you're saving a ton of bandwidth. I see it being useful for home automation. We've already got smart speakers. There was a place in Belgium that was putting audio sensors on their escalators, and they were using these sensors to identify problems with the escalators, to determine if the escalators needed repair before they actually broke down, and either hurt people or were out of action for weeks. Predictive maintenance. This idea of anomaly detection is a big thing in machine learning, especially embedded machine learning.

"It's also getting a lot cheaper, even more so now that Arm is starting to push out specific things for machine learning. Google has the little USB TensorFlow devices that you can just plug into your laptop. We're seeing more and more devices like this – cheap hardware and the ability to execute machine learning coming very efficiently together.

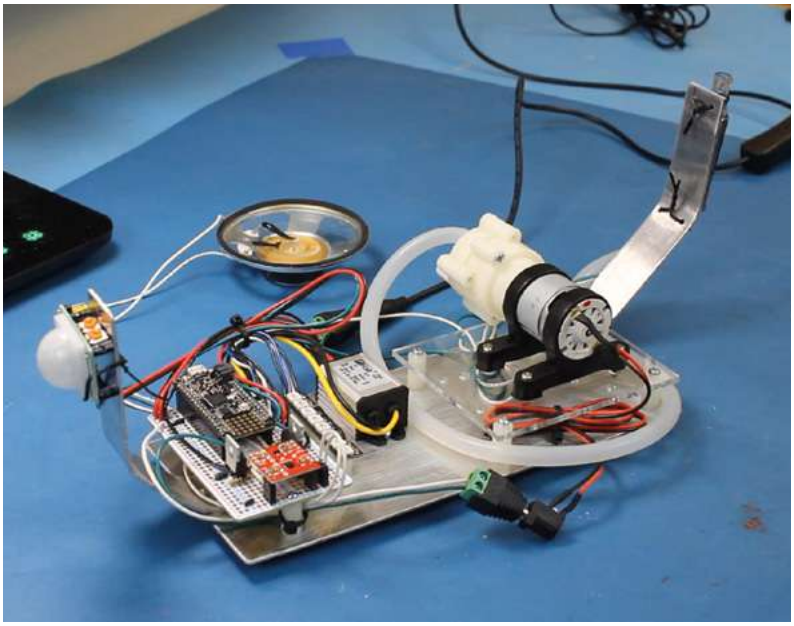
"I believe it's going to open up a new world of possibilities for us. I'd like to think it's like the early days of computing, back in the 1960s and 1970s, but I know better than that. It feels like the early days of DSP that I was too young to remember. The idea that I can perform signal processing on a digital device to do things like filtering, modulation, demodulation, audio channels, switching channels – that all used to be done in hardware. You'd have individual devices that you had to tune, and suddenly you could do it all digitally.

"In the early days of DSP, people didn't tinker with it outside of schools and industry. Whole industries bloomed around it. Texas Instruments (TI) became known as the DSP leaders when they started making chips specifically for DSP. Everyone knows TI as the DSP leader because they jumped on it as soon as they could.

"And it feels kind of like that. All these industries are coming up that use machine learning, and it's so exciting.

Right
You can tell this image is a couple of years old; the timber in that workbench would cost at least a million pounds nowadays

Credit
"SparkFun Culture" by SparkFun Electronics (CC BY 2.0)



"We obviously have stuff around AI and ML that's been around for ten years. Netflix is using ML to recommend movies to you. But the idea of embedded machine learning is newer than that. We're seeing makers jump on this. We're seeing new hardware come out – and I don't just mean the same things with a faster chip, I mean brand-new silicon put together to support ML, which I haven't seen since graphics cards and DSP chips, so that's an exciting thing in the hardware world. We really haven't seen much that's novel in the hardware world in ten years.

"It's a fascinating field, and the one thing I think is really cool is that the maker movement is getting into this because of all these content platforms and the ability to learn from and teach each other. You no longer have to have a master's degree in electrical engineering in order to understand what machine learning is, like you did with DSP 20 years ago.

"I didn't know what DSP was and I only took DSP courses as senior years' courses as an undergraduate. It wasn't until I had all this body of knowledge built up over time as an undergraduate that I knew microcontrollers, understood signals and systems, I understood the concepts, and could combine them to learn DSP. I thought that was super-slick, but at the time it was already a ten-year-old industry. Things are moving so quickly that we're now, year two, into really getting into embedded machine learning, and colleges are already teaching classes on it.

"This ability to create content and teach each other is exponentially higher than what it was, and a lot of that is down to the maker movement and its ability to teach each other.

"I love machine learning. It took something in which I was interested in college, built on it, and to me, it's opening up a new world of possibilities. I don't know where TinyML or embedded machine learning is going. The possibilities are very exciting. There could be creepy cameras watching our every move from now on, and that's a potential consequence, but there has always been that risk; technology is a tool, it's how you use it.

"I love the idea of more smart speaker things where I can interact not just through a mouse/keyboard. Give me other ways. Pull us closer to *Star Trek*. 'Hello computer' – I want to be able to talk at my computer, and we're getting there. It's exciting."

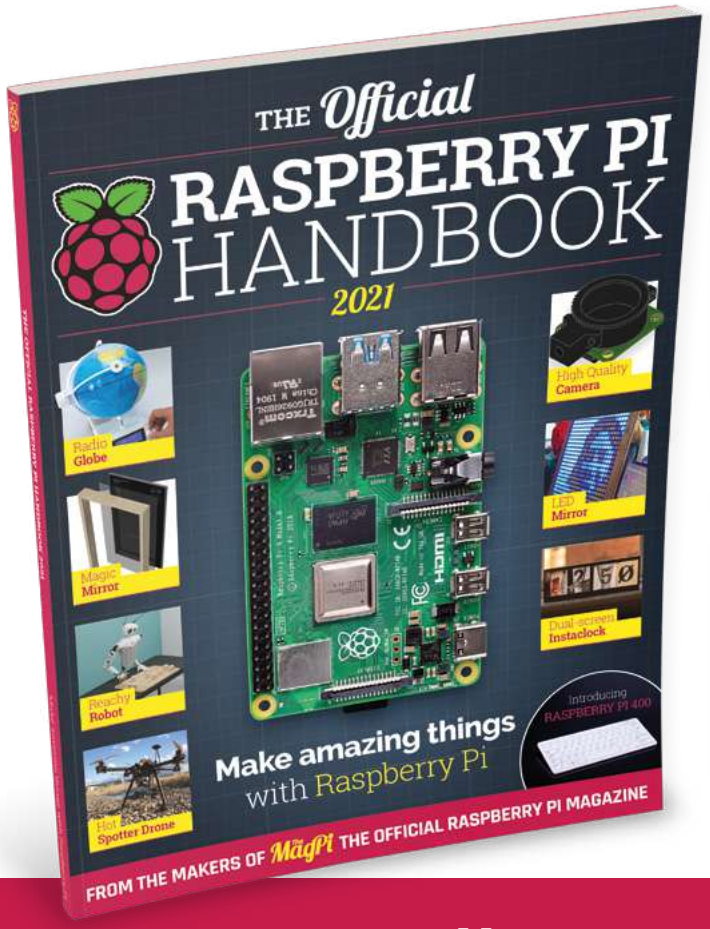
Shawn co-hosts a podcast for makers who want to learn how to turn their products into a business. It's called Hello Blink, and it's infinitely more applicable than any marketing course we've ever done

Left
Spoopy Poopy uses an Adafruit Feather M0 Express running CircuitPython to listens for the phrase "trick or treat" as a trigger to release an unpleasant smell. Perfect for keeping kids off your lawn!

THE *Official* RASPBERRY PI HANDBOOK 2021

200 PAGES OF RASPBERRY PI

- Get started with Raspberry Pi, electronics, and more
- Be inspired by incredible projects made by other people
- Learn how to code and make with our step-by-step tutorials
- Find out about the top kits and accessories for your projects



Buy online: magpi.cc/store

Hacking around the world

A brief history of hackerspaces

Drew Fustini

@pdp7

Drew Fustini is a hardware designer and embedded Linux developer. He is the Vice President of the Open Source Hardware Association, and a board member of the BeagleBoard.org Foundation. Drew designs circuit boards for OSH Park, a PCB manufacturing service, and maintains the Adafruit BeagleBone Python library.

Hackerspaces (known as hackspaces in the UK) are community-run places where people can meet up to share knowledge, learn new things, and work on projects. The mission of any hackerspace is to bring people together to collaborate on projects and educate each other. Members of a hackerspace community cover a broad terrain of disciplines, from artists to engineers; and a wide range of experience levels, from beginners to seasoned professionals. Every hackerspace has its own unique history, often intertwined with the stories of other spaces and hacker events.

Although my own relationship to hackerspaces began in the USA, the movement itself started primarily in Germany and Austria. Berlin hackerspace c-base has been in operation since 1995, and is widely regarded as the mother of all hackerspaces. Or perhaps I should say 'mothership', as members solemnly swear that their shared space is, in fact, part of the remains of a partially buried 4.5 billion-year-old space station. Entering through the spaceship portal, ordering a cold German beer or Club-Mate at the bar, and controlling the Flaschen Taschen pixel art LED installation from your phone are iconic c-base experiences, especially combined with their raucous New Year's Eve hacker parties and that killer riverside view of the Berliner Fernsehturm!

In the summer of 2007, a group from the USA called 'Hackers on a Plane' attended the Chaos Communication Camp in Germany (an incredible festival that comes around once every four years). The trip included a tour of European hackerspaces like c-base, Metalab in Vienna, and others.

Inspired by their journey, members of the group set out to found hackerspaces in their own cities. Bre Pettis helped found NYC Resistor, while Eric Michaud and Nick Farr got HacDC off the ground in Washington, D.C., and Mitch Altman followed by co-founding Noisebridge in San Francisco.

> **I hope you consider seeking out a hackerspace or two to try something new, teach something, share a story, or work on a project**

From these first few hackerspace seeds, many more grew across the country, with similar stories unfolding on a global level. By now, the hackerspace movement has spread to every continent other than Antarctica, with **hackerspaces.org** mapping more than 930 active hackerspaces, and another 362 being planned or built out. Hackerspaces and hacker culture enabled countless inventions, collaborations, and projects – including MakerBot, founded out of NYC Resistor; and makerspaces, which started coming into existence after 2005.

These community spaces are a precious shared resource. Over the coming months, as we slowly start to return to life with other people, I hope you consider seeking out a hackerspace or two to try something new, teach something, share a story, or work on a project.

Letters

ATTENTION ALL MAKERS!

If you have something you'd like to get off your chest (or even throw a word of praise in our direction), let us know at **hsmag.cc/hello**

CHOC MATE 2

According to William Gibson, the future is already here; it's just not evenly distributed. Well, you can keep your jet packs: I want a 3D printer that will create chocolate bars! Even Willy Wonka didn't have one of those!

Alberto Rodriquez
LA

Ben says: At the time of writing, I thought there would be a small niche for a 3D printer that uses chocolate. I was wrong: the choc mate 2 flew past its crowdfunding goal, and I can't wait to see it in action. It turns out that people… like… chocolate. Incredible.

INTERNET OF DIY THINGS

I've always liked the idea of the Internet of Things, but in practice, it's too close to a Cory Doctorow novel. Imagine your phone tracking your Strava runs and sending it to your healthcare provider, letting them know you've not been for that run you promised you were going to go on? Or your online shopping service being able to manipulate the price of cream to make you spend more if it knows, from your smart fridge, that you have a load of strawberries in the fridge? It could be excellent, but the future has unfortunately been hijacked by a couple of big companies that depend on advertising revenues. This is why we can't have nice things. Unless we make them ourselves: the smart fridge in issue 43 is a perfect example of what we could and should have.

Robin
Toronto

Ben says: Those are relatively benign examples; as you say, the idea of funnelling our personal data through a handful of companies is pretty dodgy. I like the idea that I can automatically add milk to an online shopping list; I like it a lot more when it's a list that only I can see. That's what open hardware is about: putting the power in your hands.

KNOTS IN SPAAAACE!

Your article on space knots got me thinking about going camping when we're allowed to. At the tail-end of the wettest May in history, I'm feeling a bit peeved that I can't even get out into the garden, let alone the great outdoors, but when it happens, I know I'm going to impress the kids with some space-age technology.

David
Birmingham

Ben says: Mirabile dictu, it's warm again in the UK, and after we've got the initial sunburn and cider out of our systems, we'll be planning camping trips again – this year armed with the knowledge that our knots are space-agency-certified.

CROWDFUNDING NOW

Inkplate 6PLUS

A big display that doesn't drain the battery

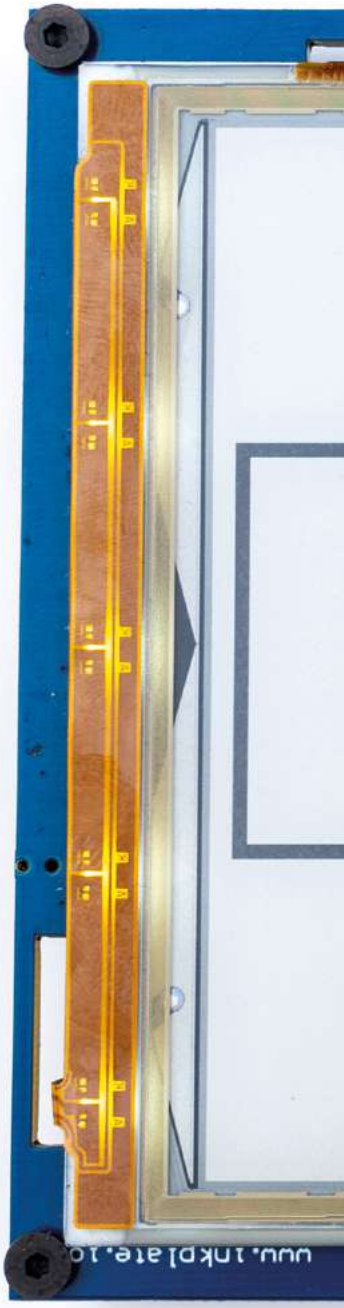
From $159 | hsmag.cc/VZJ1ng | Delivery: November 2021

It doesn't matter how clever your logic is or how fancy your sensors are; a large part of the experience of using your device will be its user interface. This might be a screen, some LEDs, or perhaps even voice control. However, if you want something power-efficient that works well in natural light, e-paper is becoming a great option.

This display technology has been around for a while now and is popular in both consumer and industrial products, but it's been a little slow to take off in the maker world. E-paper displays historically have been expensive and hard to work with, but that's starting to change now. Smaller displays are now available from a range of maker manufacturers, and larger displays – such as this one from e-radionica – are becoming more affordable.

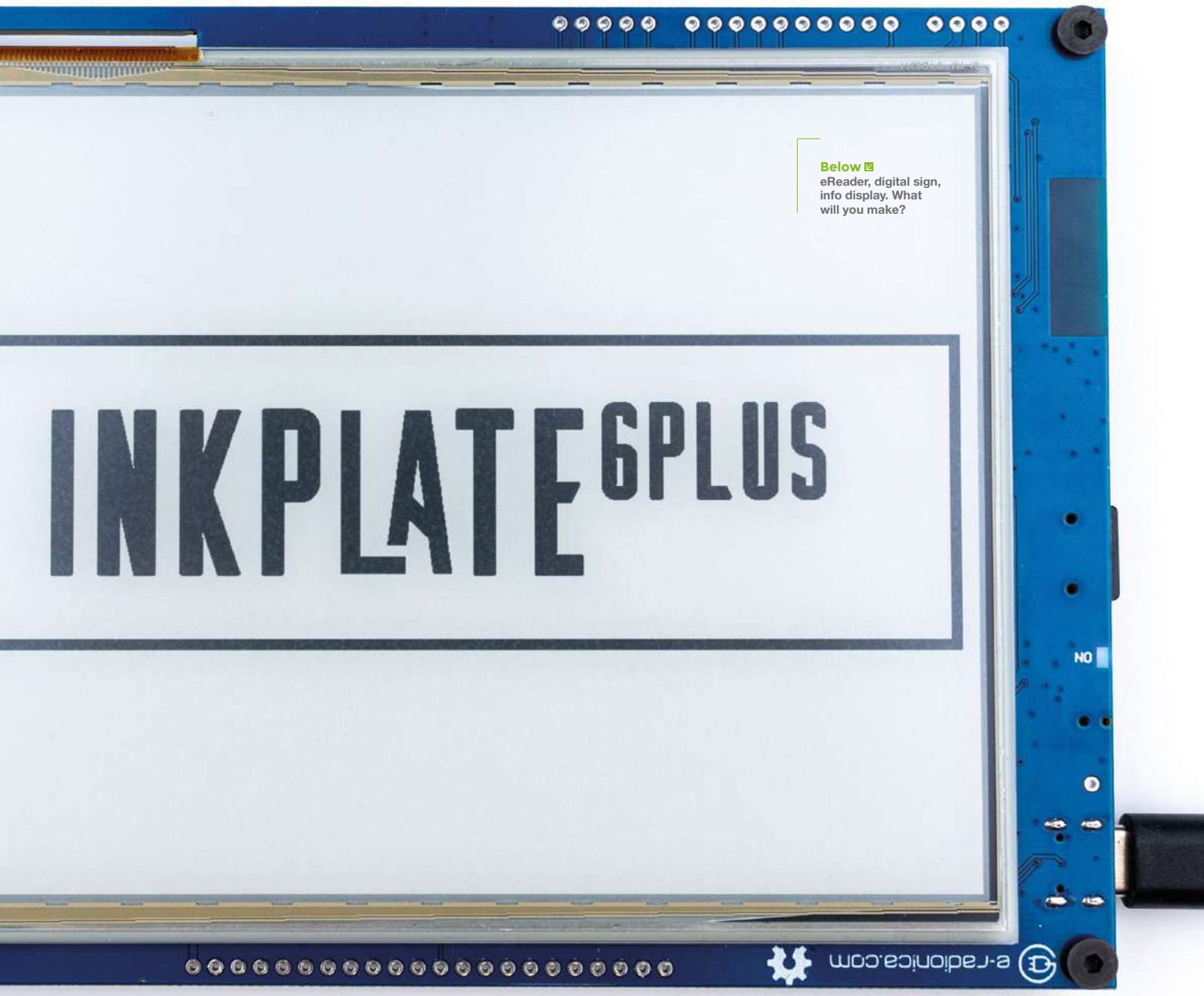
At its heart, the Inkplate 6PLUS has a recycled Carta panel from a Kindle Paperwhite (or similar) e-reader. This has a resolution of 212 PPI, and a 1.27 second refresh time (this can be decreased with partial refreshes). An ESP32 microcontroller sits alongside this display and brings in WiFi control – it also has excellent support for low-power operation. In sleep mode, it consumes just 22 µA, so it's a good choice for battery-powered applications.

We're really enjoying seeing more and more e-paper displays become available for makers. The Inkplate 6PLUS looks like it'll become a great option.



BUYER BEWARE

When backing a crowdfunding campaign, you are not purchasing a finished product, but supporting a project working on something new. There is a very real chance that the product will never ship and you'll lose your money. It's a great way to support projects you like and get some cheap hardware in the process, but if you use it purely as a chance to snag cheap stuff, you may find that you get burned.



Below
eReader, digital sign, info display. What will you make?

Build a Makerspace for Young People

Join our **free online training course on makerspace design** to get expert advice for setting up a makerspace in your school or community.

Sign up today: rpf.io/makerspace

Raspberry Pi

The Raspberry Pi Foundation. UK registered charity 1129409

#RPiLearn

HackSpace
TECHNOLOGY IN YOUR HANDS

LENS

HACK | MAKE | BUILD | CREATE

Uncover the technology that's powering the future

PG 48

HOW I MADE: WINDIY

Humble neodymium magnets get a new purpose in life: turning wind into electricity

PG 54

INTERVIEW: SIMONE GIERTZ

The queen of rubbish robots on what life is like now she's making things that are useful

PG 34

ARDUINO + PICO

Combine the two big maker ecosystems for great power

PG 64

IMPROVISER'S TOOLBOX: PALLETS

Cheap, comes in standard sizes, gives you hipster points: why pallets are the perfect building material

PG 68

IN THE WORKSHOP

Our continuing mission: to get funky sound-waves out of our Pico

MASTERING ARDUINO

Program Raspberry Pi Pico with the most popular microcontroller development environment

Arduino can be confusing because it can refer to a particular dev board, an organisation, a programming language, or an integrated development environment (IDE). In this article, we're going to be looking at the programming language and the IDE.

Historically, getting all the software you needed to program a microcontroller could be a difficult task. Arduino revolutionised the scene by bringing everything together into one IDE. Just install one application and it takes care of everything for you. That, on its own, was impressive, but what makes it really special is that they don't just support one dev board or microcontroller. They don't even just support the ones they create. They support a phenomenal range of hardware – in fact, almost every hobbyist microcontroller is supported by the Arduino IDE, including Raspberry Pi Pico. →

IDE 2.0

We'll be using Raspberry Pi Pico in this article because, if you're a regular HackSpace magazine reader or subscriber, you'll already have your very own Pico. However, if you'd rather follow along with a different microcontroller, it should work. There's no official list of microcontrollers that are supported by the Arduino IDE, but if you look at the place you got your microcontroller, it should tell you if it's supported and how to install the appropriate components.

The first thing we need to do is install the IDE and then add support for Pico. We're going to use the IDE version 2.0. This is still in beta, so you might come across some quirks, but it is the future – the previous version (1.8) is now in maintenance mode. If you're using Arduino for the first time, it's best to start with 2.0. That said, the underlying features (at least the ones we'll be using) and language are the same, so again, this should all work on an older version if you'd rather not use the beta version.

You can get the latest version from **arduino.cc/en/software**. Scroll down to Experimental Software and select the appropriate version for your platform. At the time of writing, the latest version is 2.0.0-beta.7, but this may have moved on by the time you see this. You don't have to donate to download the software, but if you're a regular user and can afford it, it's a good way of helping to ensure that it continues to get upgrades.

Once it's installed, open the application. You should see something like **Figure 1**. We need to install the extra bits that let us work with Raspberry Pi Pico boards. Click on the 'Board manager' icon (the chip in the top left) and search for 'Arduino Mbed OS RP2040 Boards'. This will download the bits we need to compile software for Pico.

Once that's installed, let's get our first program working. There's a long tradition of starting work on a microcontroller by making an LED blink, and we're not ones to shirk tradition. Go to File > Examples > Basics > Blink. This will open a new window with the Blink sketch in it.

Now, plug in your Pico, hold down the BOOTSEL button, and connect it to your computer via USB. The Arduino IDE may detect it automatically, but if not, use the drop-down at the top to select Other Boards and Ports, then search for Pico. It may tell you that it isn't connected, but this shouldn't cause a problem. Once that's selected, you can press the Upload arrow (second from the left on the top icon bar) to load the sketch to your Pico, and the LED should start flashing.

OTHER BOARDS

While we're fans of the Pico, you don't have to use this board if you don't want to. Here are a few other Arduino-compatible boards that we've tested out and enjoyed using:

Arduino Nano Every
At just nine euros, this is Arduino's cheapest offering, but it has a few tricks up its sleeve. Running at 5 V (most modern microcontrollers run at 3.3 V) means it can connect to some hardware more easily.

SparkFun Thing Plus – ESP32 WROOM
This board brings the popular WiFi-compatible ESP32 board into the Feather form factor, which means there are lots of add-on boards (known as Wings) that can be plugged straight in.

Adafruit Grand Central
This board packs a powerful M4 processor with a mind-boggling number of I/O pins to make it great when you have lots of things to connect together.

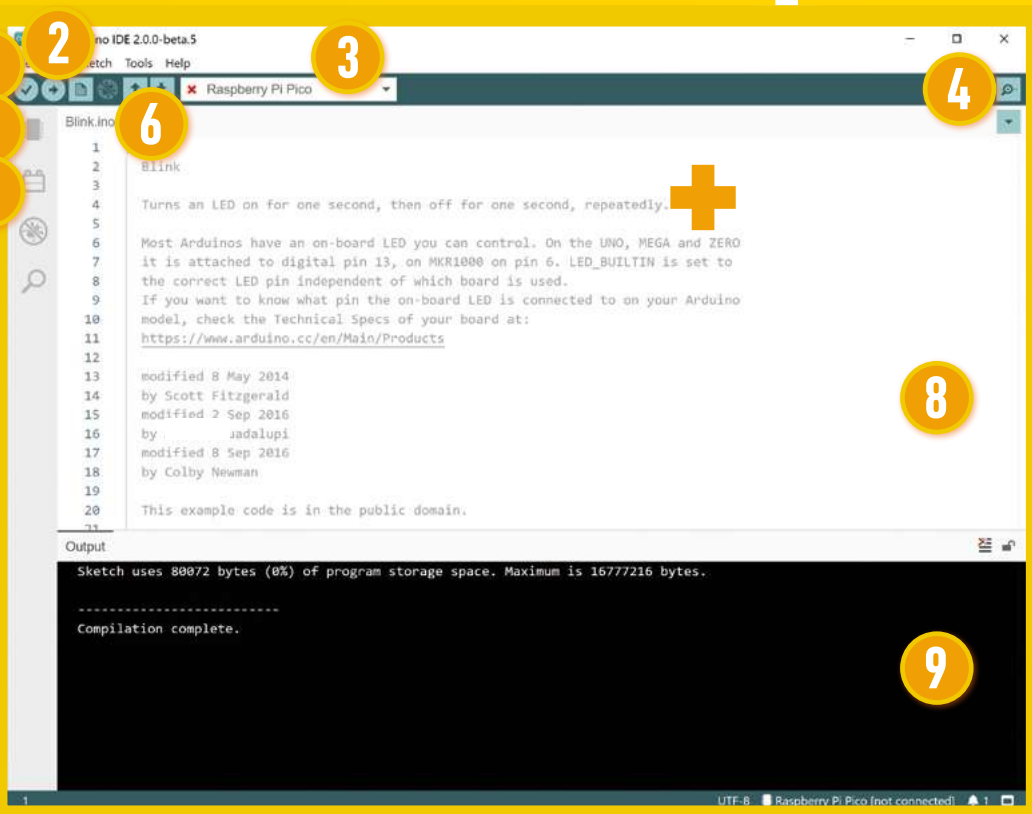


Figure 1 The Arduino IDE version 2.0 interface

1 Compile
Press this button to check that your code is syntactically correct. If it is, it will be compiled into firmware code for the selected microcontroller. You don't need to have a microcontroller attached to do this.

2 Compile and Upload
This will compile your code and then upload it to the selected microcontroller.

3 Microcontroller and port selection
This can be used to select both the type of microcontroller you want to work with and the port with which it's attached to the computer. This should automatically detect most microcontrollers when they're plugged in, but you can set it manually if you need to (or if you have more than one microcontroller attached).

4 Serial monitor
This will open a tab in the bottom pane displaying what's being sent on the serial port selected in 3.

5 Board manager
This opens a pane where you can install new board definitions (allowing you to work with particular models of microcontroller boards).

6 Tab select
An Arduino sketch may contain multiple files, and you can open them in different tabs in the IDE.

7 Library manager
The huge range of available libraries is one of the biggest advantages of the Arduino ecosystem. Click this button to open the pane, where you can search for and install the ones you need for your project.

8 The code window
This is where you enter the code for your project.

9 Output
Here you'll see any output from the toolchain that compiles and uploads the code. This could be any errors in the code or problems communicating with the board. The serial monitor will open as a tab on this pane if you open it. →

YOUR FIRST SKETCHES

We've looked at the user interface; let's now take a look at how to write code in Arduino. We'll start by looking at the Blink sketch that we used previously. The majority of the text in that sketch is comments. These show up grey in the IDE and are discarded by the compiler – they're just there to help programmers understand what's going on. If we strip these away, we get the following:

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

The language of Arduino is sort of C++. It's compiled by a C++ compiler and follows C++ syntax, but it's only sort of C++ because there are a lot of other things already set up that we can use without having to worry about them. We can't go through the whole of C++ in this article, but we will have a look at the bits that are unique to Arduino. The most obvious of these are the `setup` and `loop` functions. A sketch only needs to have these two functions to run. The `setup` function is called once when the microcontroller is powered on or reset, and `loop` will run continuously.

As well as these two, there are numerous functions that deal with common microcontroller tasks. The ones we use here are `pinMode`, `digitalWrite`, and `delay`. The first is used to set a pin to input and output, the second is used to set its value, and the final is used to pause the program by a set number of milliseconds.

As well as functions, there are defined constants that are useful to us. `LED_BUILTIN`, for example, should always be set to the development board's GPIO pin that has an LED on it. On Pico, this is GPIO 25, but it will be different on other development boards. This allows us to write code that will work on any microcontroller board without having to change the value of the I/O pin each time. Similarly, `HIGH` and `LOW` are defined to be the values of setting (or reading) a high voltage and a low voltage.

There's a complete list of all the Arduino-specific things here: **arduino.cc/reference/en/**.

Let's go back to our sketch. The `setup` function – which runs once when the microcontroller starts – sets the pin mode of GPIO 25 (the on-board LED) to output. The `loop` function then continuously runs and, each time it does, it sets the GPIO to `HIGH` (that's 3.3V on Pico), then waits for a second, then sets it to `LOW` (0V), then waits for a second.

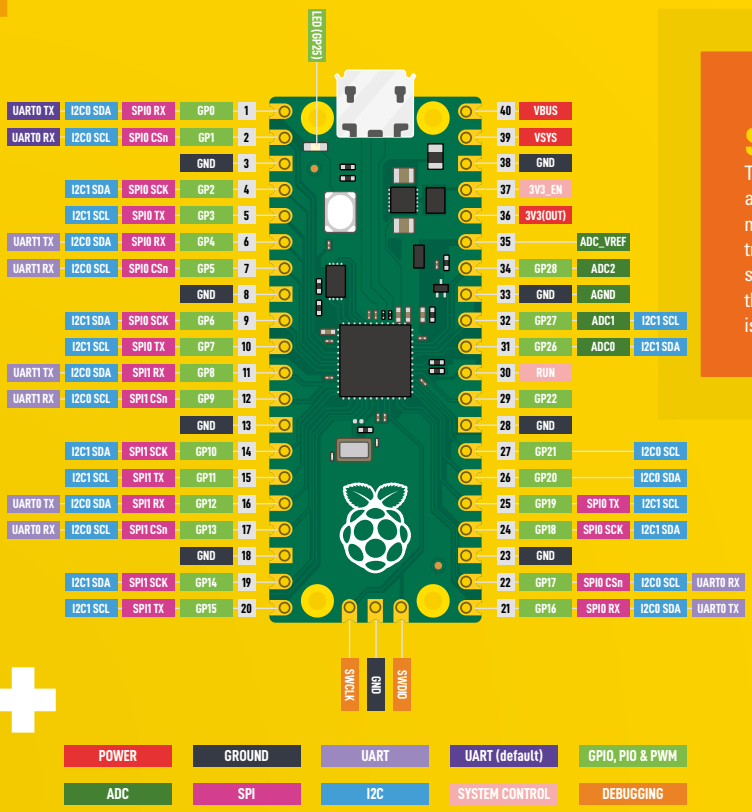
SENDING DATA

Let's take a look at another example, one that will read input from one of Pico's analogue input and send them to an attached computer. Sending data back and forth over the USB connection is a common use of microcontrollers. It's handy for testing, and also attaching sensors and other devices to our computers.

DISPLAYING THE DATA

The raw numbers coming out of the serial monitor may be a little hard to understand, but we can make things easier with a serial monitor plotter. This was a standard feature of Arduino IDE 1.8, but at the time of writing hadn't made it into version 2.0. However, this isn't a huge problem since we can use a separate program. We've used **hsmag.cc/SerialPortPlotter**, but there are others available. Download the latest release zip, uncompress it, and run **serial_port_plotter.exe**.

You can only connect one program to a serial port at a time, so you'll need to close the Arduino serial monitor before you can connect the serial port plotter.



SERIAL PORT PROBLEMS

The serial connection between Pico and the Arduino IDE 2.0 seems a little unstable at the moment. If you have problems connecting, make sure you only have one window of the Arduino IDE open and try disconnecting and reconnecting Pico. If you can't seem to get a stable connection, then you can try using the IDE version 1.8 (from the download link on the previous page). Hopefully the stability issues will be sorted in the final release.

Left
Pico's 40 pins include a host of power, GPIO, and analogue pins

```
void setup() {
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  Serial.begin(115200);
}

void loop() {
  Serial.print('$');
  Serial.print(analogRead(A0));
  Serial.print(" ");
  Serial.print(analogRead(A1));
  Serial.print(" ");
  Serial.print(analogRead(A2));
  Serial.println(";");
  delay(100);
}
```

This will run as it is, but without a sensor attached to the analogue input, the results may be a bit erratic and won't mean much. However, it does give us a chance to experiment with the code, and we'll look more into attaching hardware later.

There are three analogue input pins on Pico: A0, A1, and A2 (labelled ADC0, 1, and 2 on the pinout diagram, above). First, we set these to input, then we initialise the serial port. This serial port basically gives us a text channel between Pico and our computer, and we can send whatever text we want.

In the **loop** function, we use **analogRead()** to read the values of the three different analogue inputs. We use two different ways of sending data to the serial port: **print** and **println**. These both work in the same way except that **println** adds a new line character to the end of the output. Since we only use **println** on the final print of the loop, this means that each cycle of the loop will output one line of text. First, there's a $ character, then the value of A0, then a space, then A1, and so on until there's a final semicolon.

This might seem a bit of a funny way of outputting the values, but we've done this format so that it works with a serial plotter program (see Displaying The Data).

Printing data to serial throws the text (the numbers are converted to text before being sent) down a communication channel to your computer. This serial connection will show up a little differently depending on what operating system your main computer is running. On Windows, it's a COM port; on Linux and macOS, it's a TTY. There are lots of different programs that can read this data, and it's fairly straightforward to write your own code that reads in this text if you want to process this data in some way. The Arduino IDE includes a serial monitor that can display the text. Click on the icon in the top right-hand corner and you should see scrolling lines of output. See the box 'Displaying the data' for one way of converting these numbers into a moving graph.

These two sketches may have both been simple, but we've explored some of the key ways of using Arduino-compatible microcontrollers. By using these techniques of outputting data, reading it in, and sending it over a USB connection, you can create really powerful devices. On the next page, we'll expand on this with a useful little project. →

FEATURE

A COLOUR-CHANGING LAMP

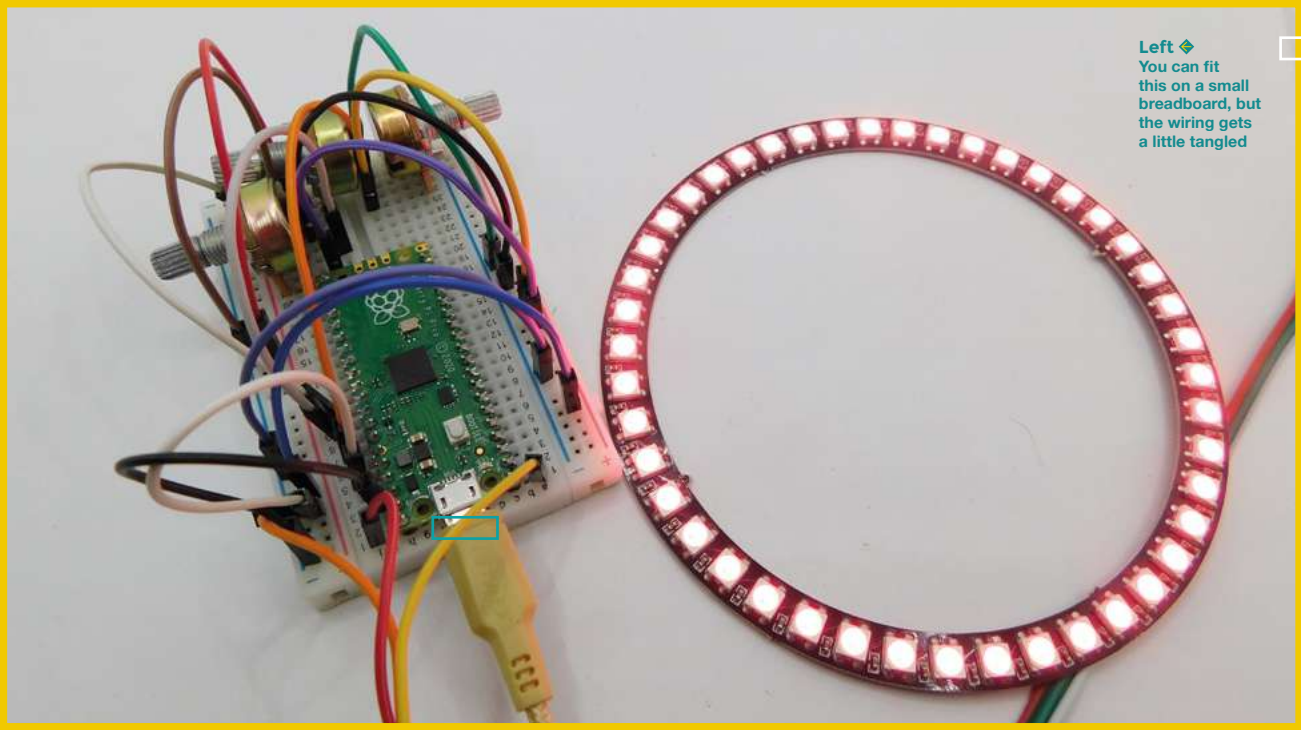
We've now seen how we can program Pico, and set digital pins, and read analogue pins. Let's take a look at how we can use this to create a simple project: a colour-changing lamp.

The concept is simple. We're going to use a ring of addressable LEDs and three potentiometers to create a light that we can change the colour of. The ring means it will provide a soft light for photography. We're using a ring of 45 WS2812B LEDs. These are available from multiple sources and allow us to control a lot of LEDs using just a single I/O pin.

We'll use the three analogue inputs to control each of the three primary colours (red, green, and blue), so by twiddling the knobs, you can change the colour of the LEDs.

Potentiometers are components that let you alter a voltage level. Connect one end to the maximum voltage you want (in our case, 3.3V), the other end to ground, and then the middle connection will vary between these two extremes depending on where the knob is twisted to. See **Figure 2** for a wiring diagram of this.

You should notice that the values change from 0 (or very low) to 1024 as you twiddle the knobs.



Left
You can fit this on a small breadboard, but the wiring gets a little tangled

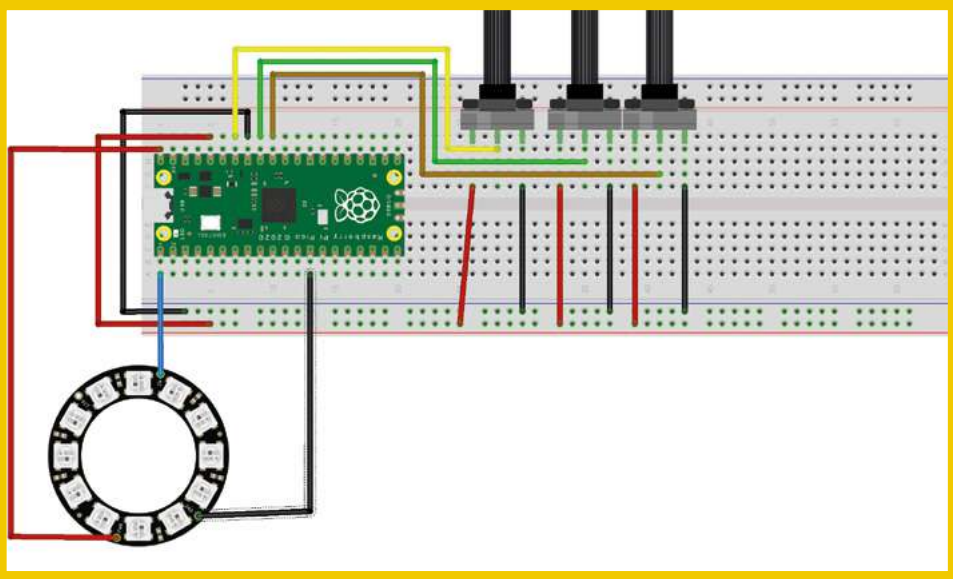


Figure 2
Your WS2812B ring may have its connections in slightly different places to this, but it should have Data In, Power, and Ground

MORE NEOPIXEL FUN

We've only scratched the surface of what you can do with WS2812B LEDs here. We highly recommend you spend a little time playing with the examples that come with the NeoPixel library. Go to File > Examples > Adafruit NeoPixel to see the options and have a play. You can create some really impressive effects with just a few lines of code.

WS2812B LEDs are sometimes known as NeoPixels as they're sold under this brand name by Adafruit. We can control them using the Adafruit NeoPixel library. Click on the building block icon on the left-hand side to open the library manager and search for Adafruit NeoPixel. You'll need to install the library that's just called 'Adafruit NeoPixel'. This gives us some functions and defines to help us work with these LEDs. Once that's installed, the following sketch should compile. You'll also need to connect your NeoPixel ring. Power should go to 5V (labelled VBUS on Pico), ground to ground, and Data In (sometimes marked DIN on the WS2812B board) to GPIO 0.

```
#include <Adafruit_NeoPixel.h>
#define PIN 0
#define LENGTH 45

Adafruit_NeoPixel strip = Adafruit_NeoPixel(LENGTH,
PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
}

void loop() {
  strip.fill(strip.Color(int(analogRead(A0)/4),int(
analogRead(A1)/4), int(analogRead(A2)/4) ));
  strip.show();
  delay(100);
}
```

In this sketch we're adding two #defines. These are ways of setting values so they can be easily changed in the code, but are fixed once you compile the program. In this case, you can change the NeoPixel Pin and number of LEDs in the ring using **PIN** and **LENGTH**.

Notice that we create the NeoPixel strip outside the **setup** and **loop** functions. We can create variables here (and **strip** is an object), but you can't run arbitrary code here.

In the **setup** function, we declare all our inputs as we have done before.

The first line in the **loop** function does a few things. The **strip.fill()** function sets all the LEDs to a single colour (you can also use the **strip.setPixelColor** function to set an individual LED colour). This function takes a single parameter that's a colour type. We can create a colour type using the **strip.Color** function that takes three parameters, one each for red, green, and blue. These should each take a value between 0 and 255, so we have to divide the output of the **analogRead** function by 4. Since we have to send whole numbers to the **strip.Color** function, we use the **int** function to convert the values to whole numbers.

Phew. That's a lot for one line, but it's also almost all our code has to do. The remaining two lines update the LEDs and delay for a tenth of a second.

That's all there is to it. Upload that code to Pico and you'll have your very own colour-changing light. →

ARDUINO PROJECTS WE LOVE

Some inspiration for your next build

The Arduino platform has been the platform of choice for makers using microcontrollers for over 15 years. In that time, there have been some fantastic projects built around a huge range of different microcontroller development boards that are supported by the IDE. Here are a few of our favourite projects.

PRUSA i3 MK3S

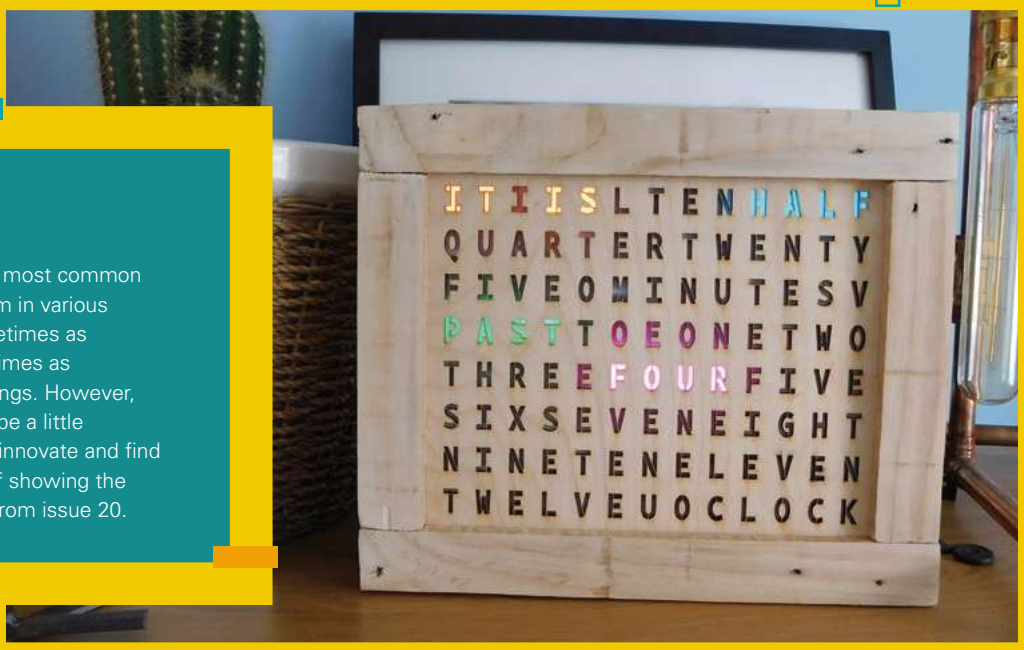
We commonly think of Arduino-compatible devices being microcontroller dev boards, but there are also projects built off these dev boards. Many 3D printers, including this author's day-to-day printer, a Prusa i3 MK3S, are Arduino-compatible. This isn't just hitting marketing buzzwords. Having an easy-to-set-up development environment helped foster a large community around 3D printer firmware, which made it robust and able to support a wide range of different hardware.

MIDI CONTROLLER

Music is a fantastic area for makers to play in. It's easy to get started, and with a little tinkering, you can create some unique sounds. There are many ways to go about it, and one popular one is to create a gadget for controlling MIDI devices. This one by John Park uses an Adafruit NeoTrellis to create sequences of notes that your computer can then turn into beautiful sounds. →

THIS ISN'T JUST HITTING MARKETING BUZZWORDS

FEATURE



WORD CLOCK

Clocks are probably one of the most common electronic items. We have them in various forms all over the place – sometimes as standalone devices and sometimes as functions on more complex things. However, the interfaces can sometimes be a little lacklustre. As makers, we can innovate and find beautiful and inventive ways of showing the time, such as this word clock from issue 20.

FIBONACCI LED DISPLAY

LED displays are common, but they almost always have a grid of pixels, or sometimes a circle. The Fibonacci256 by Evil Genius Labs LLC has the lights arranged in a Fibonacci pattern. This can be used to create swirls and animations that have a more organic look than those created by other arrangements.

ROBOT

In issue 19, we looked at how to build a four-legged walking robot. You can program this little creature to stroll around your house and, well, do whatever you want it to. Robots come in all shapes and sizes, and this is one of the smaller ones around, but it's easy to get started with just an Arduino Nano, a few servos, and a 3D printer.

ROBOTS COM
ALL SHAPES
SIZES, AND TH
IS ONE OF THE
SMALLER ONE

ROTARY CELLPHONE

One of the great things about Arduino is that it allows makers to add a new brain to existing technology. That's exactly what Justine Haupt did with the Rotary Cellphone. There's something enchanting about stripping a phone back to its most rudimentary form, yet still being able to access the mobile phone network.

CHARTREUSE

You may think of Arduino as a device for techies, but actually, it began life as a programmable board for artists. By stripping back the technology and making it as accessible as possible, it was designed to make it easy for people to add interactive elements to their artwork. This piece, named Chartreuse, will look at you as you walk past and be sad when you leave.

OPENDOG

If you want to push your robotics skills to the next level, James Bruton has designed, built, and open-sourced the fantastically complex OpenDog. This robo-mutt can trot and even obey simple hand commands. You can see all the details that went into the creation on James's YouTube channel (**hsmag.cc/JamesBruton**), or grab the CAD files and code from **github.com/Xrobots**.

FEATURE

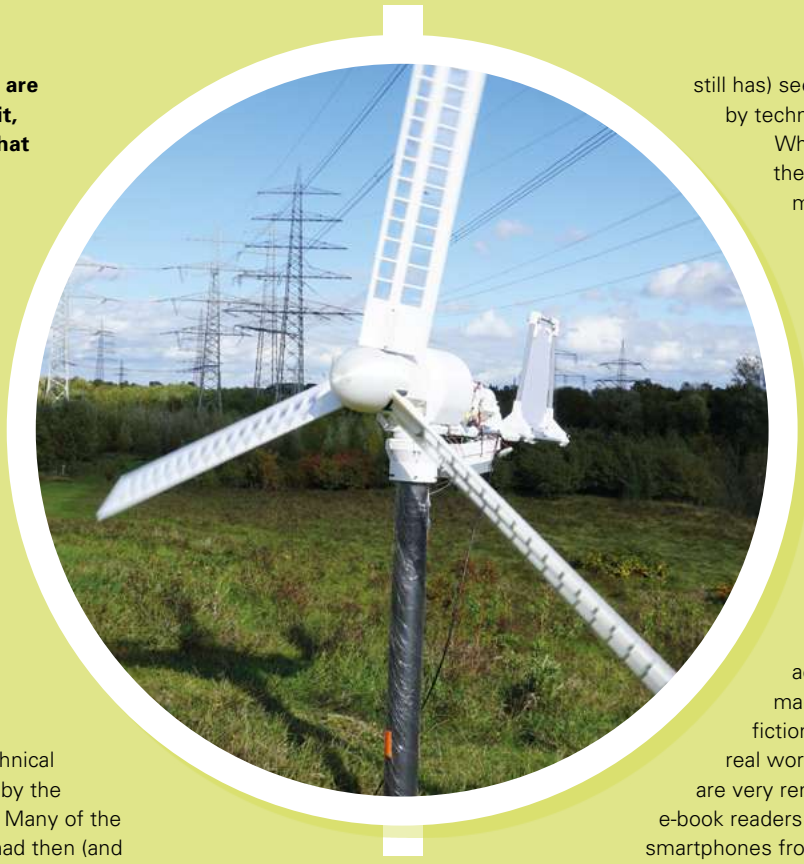
How I Made

WINDIY – A WARP CORE FOR EVERYONE

3D-print your very own wind turbine

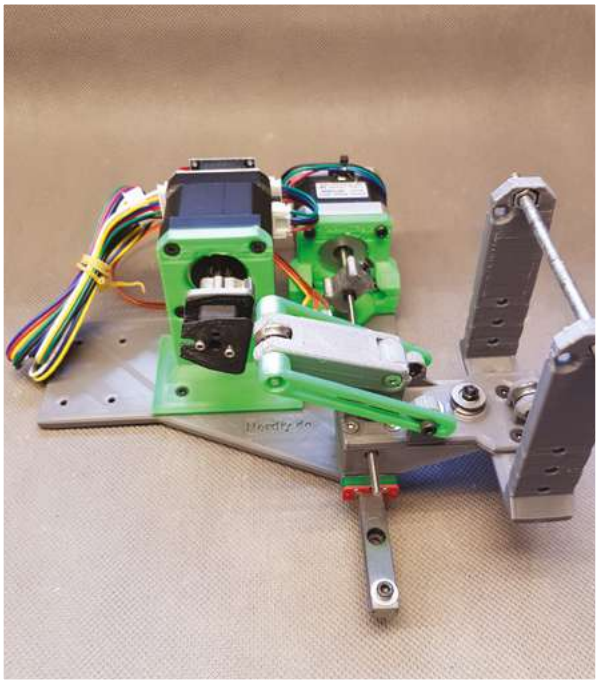
By **Fabian Steppat**

I don't know if there are studies that prove it, but I can imagine that for many people, the foundation for scientific interest is laid by science fiction. At least for me, this 'science fiction foundation' was laid by the series *Star Trek: Voyager* and my sister. As a part-time Trekkie, she regularly blocked the only TV in our parents' house with this series. In the beginning, I was forced to watch it rather than wanting to, but at some point, I watched it voluntarily and enthusiastically. I was fascinated by the many technical possibilities and, above all, by the positive view of the future. Many of the problems that our society had then (and still has) seemed to have been solved by technical progress.



Where today we leave the carbon footprint of a mammoth to fly from A to B, on *Voyager*, you were simply beamed. Nuclear or coal-fired power plants? There was no need. Instead, there was the warp core, which generated energy from matter-antimatter reactions for an infinite time without CO_2 emissions or nuclear waste.

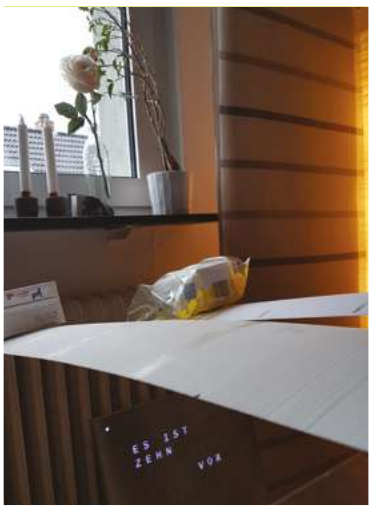
Even if this is (currently) rather utopian, there are other achievements that have made the leap from the fictional *Star Trek* world into the real world. For example, PADDs are very reminiscent of tablets or e-book readers. The majority of modern smartphones from today have a similarly



Above
Coils wound and ready for assembly

Above
The winding machine made it much easier to create the coils needed for the generator

> But the coolest of all technical devices for me was always the replicator



developed 'sensor phalanx' as the tricorders. A smartwatch is actually just a communicator that you wear on your wrist, and I think it's no coincidence that you can change the wake word of Alexa and co. to 'computer'.

But the coolest of all technical devices for me was always the replicator. You only had to tell this device – of which you could usually only see the output shaft – what you would like to have created. No matter whether it was a glass of champagne, a bowl of 'plomeek soup', or a 'hydrospanner'. Everything appeared as if by magic as soon as you told the replicator.

So, I was excited when I realised that 3D printers were becoming cheaper, more available, and more sophisticated. Even if you can't replicate food with them, you can now reproduce components with millimetre precision. At that time, many people spoke of the beginning of the second industrial revolution, and I believed in it as well. Even if the 'big bang' did not happen, I believe that there will be a time when it will be as normal to own a 3D printer as it is to own an inkjet printer today. Spare parts and necessary equipment for any device can then be made available quickly, even in regions where there is no 24-hour, door-to-door delivery service. However, for this to work, it is important that these easily replicable devices are designed to be not only easy to build, but also easy to repair.

At this point, I would like to claim that I wanted to actively and specifically support this development with WinDIY from the very beginning. In the end, however, it was actually coincidence that nudged me onto this path.

The coincidence came in the form of a few cheaply bought neodymium magnets at the flea market. When I got home, they lay in the drawer for a long time, or were used as fridge magnets. At some point, I realised that this was not a life worthy of a neodymium magnet. These powerful magnets should have more use than holding postcards on the fridge door.

So, I wanted to build something useful with them, and quite quickly came up →

FEATURE

with the idea of using the magnets to build a generator. After all, what could be more useful in a society where we need a lot of energy, and the demand is growing every day, than generating energy? Let's put it this way: if I were a neodymium magnet, I would love to be part of such an important task.

To cut a long story short, this is how the 'Nerdiskerator' (=NERdiys DISK genERATOR in reference to my blog **Nerdiy.de**) was created: a double-disc generator consisting of 20 neodymium magnets in each of the two rotor discs, and twelve coils in the stator. I made the housing parts with the help of my 3D printer, and finally assembled them with a few standard mechanical parts. For me, the 3D printer was the only suitable tool because it was the only way I could create components, test them, change them if necessary, and produce them again with consistent quality and accuracy.

Since I wasn't really talented at working with CAD tools at that time, I printed every design and tested the physical part. Pretty soon, I had a set rhythm consisting of:

- **Evening/night:** print design
- **Midday:** test design and implement changes
- Repeat

So, night after night, I worked my way to the final design until I had a design I was happy with. In retrospect, I have to admit that this way was expensive and not always effective. Unfortunately, I only learned later that modern CAD solutions also allow you to assemble the individual components on the computer and check whether everything fits together. At that

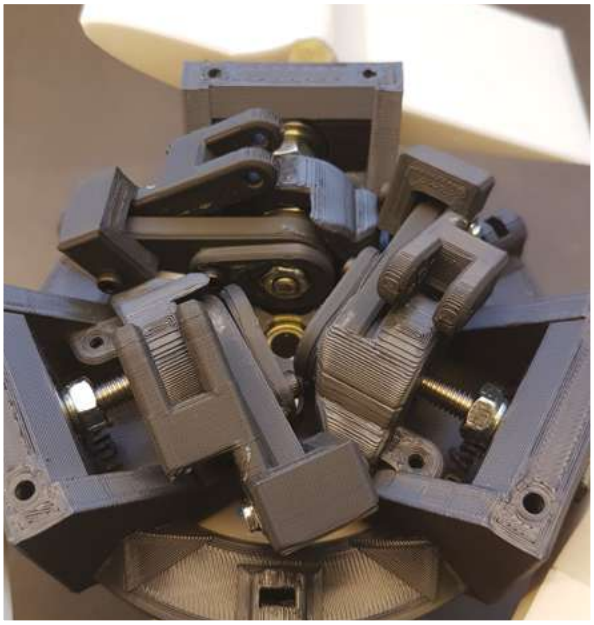
time, however, it was a good system for me to see and understand how the individual components work together.

The next challenge – after building the larger generator parts – was making the coils. Here, a lot of wire had to be wound around a coil body as evenly and accurately as possible and with a given number of windings. I quickly gave up the first attempts to do this by hand. There are machines that do this job much faster and better. Unfortunately, these cost more than I was willing to pay at the time. Hence, I decided to build my own winding machine.

'WinDIYngThing' (that's the name of the machine) consists of two NEMA17 stepper

Above
turbine rotors have to be both light and strong

Right
As well as 3D printed parts, there's some off-the-shelf components like bolts and nuts



Right
3D printing lets you quickly iterate your designs

Below
Wiring up the control electronics

motors: one is responsible for guiding the winding wire, and the other for the actual winding. So, while the winding motor winds the wire onto the coil body attached to its shaft, the second stepper motor guides the wire via a linear rail so that it's wound onto the coil body as parallel as possible.

Thanks to 'WinDIYngThing', the winding worked quite well, so that soon all twelve coils were produced and installed in the stator. During the first tests of the ready-built generator, driven by a cordless drill as a mechanical energy source, 30W could be generated. Unfortunately, it was not until later that it became clear that I should have invested in a tachometer at this point.

With the completed generator, I was now faced with a similar problem as before with the neodymium magnets. Instead of the individual magnets, the assembled generator was lying uselessly in the drawer. So, this one also called for a useful purpose. I needed a mechanical energy source to drive the generator. I chose wind power, simply because, where I live, wind power is more available than water power.

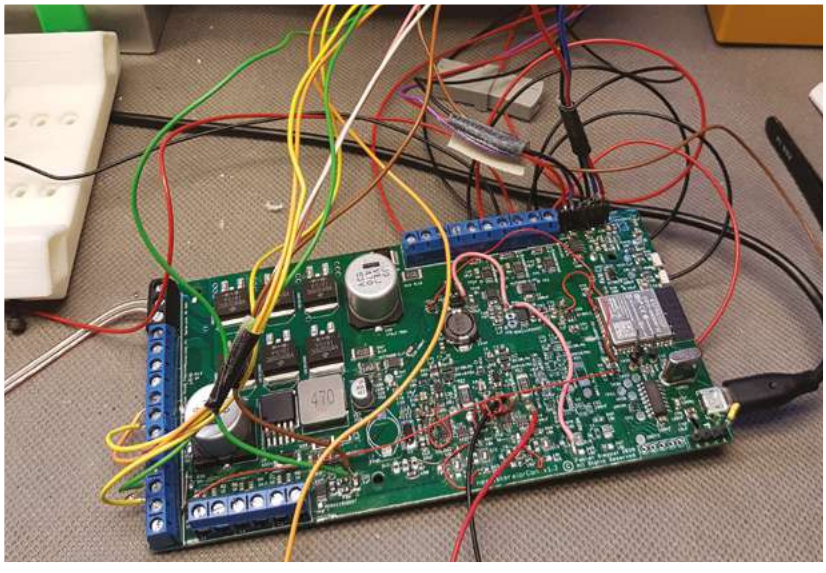
Unfortunately, I started developing and building the turbine rather haphazardly because I thought I knew how a wind turbine worked. It was only later that I realised how many, at first glance unimportant details had to be right in order to turn a decorative turbine that was simply supposed to rotate into an effective, energy-generating turbine.

At first, however, I was also concerned with the challenges of general construction, especially with regard to 3D printing. For example, the blades of the turbine must be as light as possible, but at the same time strong enough not to be damaged by the forces that occur. Various tests with different designs finally led to a blade design in which a blade weighs about 600 grams, but is still able to withstand the forces that occur. For this purpose, the wing is supported by an internal aluminium profile onto which 3D-printed wing segments are attached and fixed.

In addition, I wanted to integrate a few safety systems that would enable WinDIY to react as autonomously as possible to strong winds, or other potentially dangerous events.

Strong wind is actually a great thing, because more wind obviously leads to more energy that can be 'harvested'. On the other hand, it also increases the load on the components. There are quite impressive videos of wind turbines that reach such high speeds due to failed safety systems and too much wind, that they literally destroy themselves.

I wanted to avoid this scenario for WinDIY in any case. One way to do this is with a mechanism that can be used to adjust the angle of attack of the blades. To do this, a pushrod is passed through the axle of the generator and into the hub of the turbine. By pushing this pushrod back and forth, the angle of attack of the blades can be adjusted. By thus adjusting the angle of attack appropriately (or not appropriately), the efficiency with which



FEATURE

the wind-speed is converted into rotational speed of the turbine can be controlled.

So, in case the speed is too high, the pushrod is simply controlled that way that the turbine brakes itself. This system is currently also the common procedure of the 'big brothers', which are known from the coast or other windy regions.

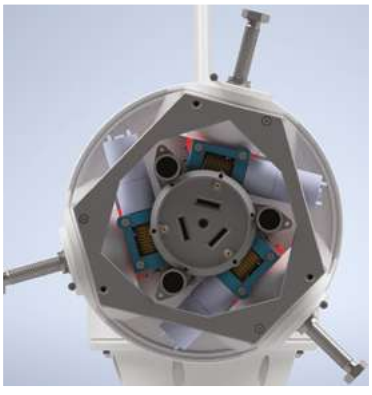
In case this is not sufficient, or the turbine has to be stopped for other reasons, a mechanical brake is installed. With the help of two electric motors, including the corresponding gear reduction, two bicycle brake-blocks can be pressed onto the rotor disc of the generator, thus bringing the entire turbine to a standstill.

The control electronics inside the turbine automatically decide whether and which safety mechanism will be activated. Equipped with an ESP32, it collects, stores, and evaluates the various sensor data and acts autonomously.

In the meantime, I had also made various changes to the design of the 'Nerdiskerator'. Having to test the generator every time with the battery drill was not only inconvenient, but also dangerous. I had to hold the drill with one hand while holding the fixed part of the generator with the other. At the same time, I also wanted to operate the multimeter. After I had dropped this construction a few times, I decided that a small test stand could manage this task better and more safely.

Left
Testing the performance of the generator

Below
Modern CAD systems let you see how multiple parts will fit together



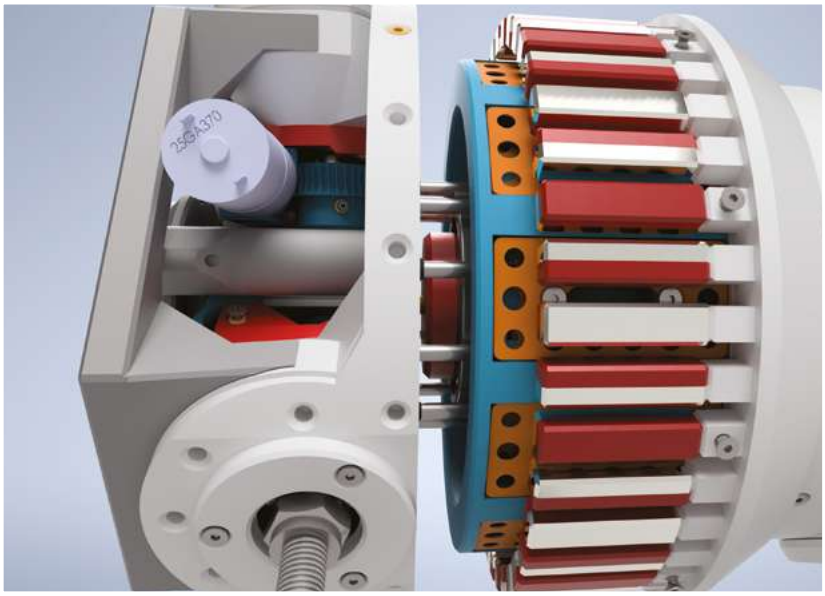
The generator only produces significant energy at about 20 revolutions per second(!). A speed that WinDIY never reached

Since there was no device that suited my budget and purpose, I decided to build my own test stand. It consists of a 200W electric motor, which is actually sold as a drive for electric scooters. Thanks to a suitable power supply and a PWM speed control, I was able to build a suitable test stand with which I was able to test the generator.

A tachometer connected to it showed the first lesson I had to learn. The generator only produces significant energy at about 20 revolutions per second(!) – a speed that WinDIY never reached even on the stormy test days and would probably not survive undamaged.

When I started building WinDIY, I underestimated that it is not enough to know roughly how a wind turbine works. The details to make a turbine really efficient are based on the knowledge of electrical engineering, mechanics, aerodynamics, and many other technical fields that should not be ignored.

At first, this realisation frustrated me quite a bit. I had invested many hours in the development and construction of the turbine, produced countless failed and test prints, and invested the corresponding filament and money. Realising then that the machine I had built worked in principle, but fell short of expectations, was pretty devastating.



Left
The current design for WinDIY 2

In this respect, I know today that the first version of WinDIY is a good example of what you can do wrong at building a wind turbine. But at the same time, I realised that WinDIY is also a good example of what is possible with a simple 3D printer, motivation, and a good search engine. Most of the knowledge on this is publicly available and free of charge. We just need to take it and turn it into something useful. This, a suitable CAD tool, and a 3D printer are tools that are already available today and make this possible for everyone.

Of course, you can't replicate a finished wind turbine with a 3D printer alone, as you would expect it from a *Star Trek* replicator. But you can make components so precisely that even inexperienced people are able to assemble a well-functioning machine from it. It is precisely this goal of a wind turbine that is easy to manufacture and rebuild that I continue to pursue with WinDIY and its successors. Even though this did not work 100% with the first version, I'm working to get closer to this goal with a new version.

In particular, the generator has to be designed in such a way that it produces significant energy even at low speeds. To ensure that enough mechanical energy reaches the generator, the wings must be equipped with a suitable profile. Besides these obvious improvements, the construction must also become simpler and easier to repair.

I want to implement all this in the upcoming version 'WinDIY_2'.

WinDIY is not (yet) a warp core replacement that generates energy from matter-antimatter reactions. But hopefully, one day, it will become a machine that effectively generates energy from wind-matter reactions on a small scale and can be built and repaired by anyone who has a 3D printer next to their inkjet printer. □

HackSpace magazine meets...

Simone Giertz

Inventor, robotics enthusiast, and YouTuber

Words **Alex Bate**

Even if you don't follow Simone Giertz on social media or YouTube, there's a good chance you know of her work. Originally hailed as the Queen of Shitty Robots, Simone's early videos of questionable contraptions, such as the Toothbrush Machine and Hair Washing Robot, quickly went viral, birthing a variety of GIFs and shareable content that quickly took over the internet. But, nowadays, she's shelved her bots and focuses her attention on more reliable projects, such as her highly successful crowdfunding campaign for The Every Day Calendar, and the impressive Truckla, a Tesla pickup truck that beat Elon Musk's Cybertruck to the post when shared online in June 2019.

Alex Bate caught up with Simone Giertz (pronounced Yetch, not Gerts) to discuss how she went from unreliable robots and GIF stardom to bunk-beds made of leaves and office chair sidecars for needy pets, and why her openly discussed brain tumour helped to realign her business model. →

HS To me, as a viewer, it feels like your YouTube career is split into two halves. There's Simone, the Queen of Shitty Robots, and then there's everything post-surgery, like Truckla and The Every Day Calendar. Do you see it too?

SG The difficult part about YouTube, and also the good side of it, is that if you have a really long career, you grow up during that career, and you change and your interests change. And I don't want to just play a role, I want to be genuinely excited about the things I do – you get sick of things, and you want to explore new things. So, in order to do that, I've really tried to be 'theme agnostic' for my YouTube channel.

And that was something that was really hard with Shitty Robots, because it was something that I knew that people really liked, and that I had a level of success with. But I was just not that excited about it anymore. And I think the brain tumour became a really good page turner for me, because I had such limited energy capital, you know, and I really just wanted to spend my time and my very limited energy on doing things that I was super-pumped about.

I think the projects I build now still have some elements of the stuff I did in my early days, but they're definitely less GIF-compatible.

In the beginning, all I was thinking about for every project was a GIF. That was the main deliverable that I had in my head, and the main piece of content that I focused on, and then I kind of built a YouTube video around it, and around the process of creating this GIF. And I let go of that. Not every project needs to have a punchline. It can be fine. It can be a little bit more dull.

But, I still feel guilty about it.

HS Really?

SG Yeah. People are very sweet about it, but I still get comments with people being like, 'Oh, I miss the Shitty Robots.' But, at the same time, you have to think, 'It's my life, and I really want to do the things I want to do.' And I'm also so drawn to my product business and wanting to focus on that. And the way that my YouTube channel can co-exist with that is for me to explore different products and make videos about them. And it's actually becoming a pretty good tag team.

HS Talking about your product business, the biggest one to date was obviously The Every Day Calendar. 2300-odd backers, and over half a million dollars raised. How did you feel when your first Kickstarter just soared like that?

SG It was fun and scary. Because, as somebody who's terrified of disappointing people, crowdfunding campaigns are kind of like the worst position to put yourself in because you really risk disappointing people. But, I don't think we did. I mean, we were late, but I really just wanted to deliver a good product because it was expensive. And, yes, we raised over half a million dollars, but it's not until now that we've actually broken even.

> Yes, we raised over half a million dollars, but it's not until now that we've actually broken even

HS Wow.

SG It's so expensive. And so much of that is in product development. When it comes down to it, and you're actually putting something out in the world, it's just crazy how much it costs. And I mean, we probably didn't do it in the most efficient way we could, because we were rookies. But, it was definitely very humbling and terrifying.

HS Would you do further products with Kickstarter? Or do you think you're now at a point where you would just create a product and sell it, and not have to rely on crowdfunding?

SG We're hopefully launching our store this summer, and we're going to have four different products in it. And, I'm hoping that any easier products can be self-funded. And, if there's something more complicated, like the Companion Chair, which is definitely going to be a bigger project, it might end up being crowdfunded because with funding, you also get market testing. You can get a lot out of it. But, that said, after I did The Every Day Calendar, I remember saying I'd never do it again. Every night at 3 am, I would just wake up and be like, 'Oh my god, what if we send out the calendars and then, in two years, all of them stop working! People are going to be really angry.' I'm scared of that. But, I guess that also, even if customers are buying your product off the shelf, you are always going to live with that fear over your head.

HS It's really interesting to go back and watch your earliest videos, particularly the first one in Swedish, and see how far you've come. Was it always the aim to start the business? To have staff and be opening an online store and selling your products?

SG I mean, no, I would definitely be lying if I said that this is some sort of master plan. There was no scheming where I had the large whiteboard – 'This is the trajectory of how I'm going to become known as the Queen of Shitty Robots. And then I'm going to pivot that into running a product business.' I'm definitely not that smart.

But, I had an inkling of what I was interested in. And I mean, I really liked making videos. And I think that

everything kind of happened in a very fortunate way. Because I had this job where I was a Maker in Residence at a US company called Punch Through Design. And my job was just to build different things. And right when my job there was ending, I posted the Toothbrush Helmet, and that started getting some traction. I was moving back to Sweden because my visa expired, and I just had this year of living with my mom again, and having very few expenses and I was like, 'OK, I'm gonna just make sure I work enough to get by, but then the rest of the time, I'm just gonna spend it on building these machines that I want to build.'

So I was very fortunate in the way that I could structure things so I was able to spend time on my YouTube channel in the early days.

But, it's also so easy to look back and be like, 'Of course, all these things led me to where I am today.' But when you're in the middle of it, you're just flailing. And my flailing, fortunately, landed me in a position that I'm very happy with today.

[It's at this point of the conversation that Simone's three-legged canine sidekick, Commander Scraps, decides to join us. Those who have seen Simone's build video for the Companion Chair or Lego-based Dog Selfie Booth will already know of Scraps. Those who haven't, well, Scraps is adorable, so you should definitely check them out.]

Right
Without the pressure of having to please the internet, Simone's on a creative roll

Above
2305 backers paid $593,352 to get the Every Day Calendar off the ground

HS Some online content creators are often stuck within a theme – wood working, electronics, 3D printing, and so on. But, for you, it seems that you're the theme, you're the brand, and you can get away with creating whatever content you want. Do you see that when you interact with your community? That freedom?

SG It's something that I thought a lot about in the early days, like, how much is the channel about me and my life? And how much is it about the things that I build? And I think what I struggled with is that I'm not that interested in my life. Like, I really want to make videos that I myself would want to watch. I'm not really interested in vlogs, so I decided early on that while it's about my life to an extent, it's still centred around these projects I'm building.

In some ways, I'm pretty private on the internet, but also very open, like when it comes to brain tumour stuff. I was really open about it, and I wanted to tell everyone about it because it was a way for me to process what was happening. I remember having to tell myself that I had to stop telling waiters or Uber drivers that I had a brain tumour. 'Hi sweetie, how are you today? Well, I have a brain tumour, but other than that, I'm pretty good.'

When it came to talking about it online, it was a no-brainer. Haha.

But then there's other stuff that I don't talk about, like, I don't really document my life. I don't talk about my friends really, or my relationship status, or anything like that. Because you have to draw the line somewhere. And I always felt like documenting my life was just too intrusive.

HS When you look at your most popular videos on your channel, even though you're known as the Queen of Shitty Robots, those videos aren't actually in the Top 5. Instead, it's the video of you in the zero gravity simulations, and Truckla, and locking yourself in your bathroom for 48 hours. It's interesting that the thing you're most known for isn't the thing your audience is most interacting with.

SG Those Shitty Robot videos mostly did really well on other platforms, like Twitter and Reddit. Not so much on YouTube because it has its own metrics and algorithms.

The thing that is really useful for other creators who are getting started is to figure out what is your hook, or what is the very simple version of what you're doing. Like, Queen of Shitty Robots kind of became the headline. And it was this very clear message, and it was something that was really easy for journalists to write about. It was a spearhead for branding.

This was not something I was thinking about at the time, but looking back, my fear then was to make sure I didn't get pigeonholed, and that I could never move on from it, because that's the problem when people only know you for one thing – you can't really move on beyond that. It's really nice to have that spearhead, and then you can broaden it, and that's how you have longevity.

> "There's nothing more inspiring than someone being pumped about something"

I didn't want this to be over in a year. I wanted to be able to keep on doing it because I was really enjoying it. And now, I want to make sure that I have more legs to stand on, because when you're going through health problems, you realise that if you can't be in front of a camera, everything grinds to a halt. If you're not well enough to work, or if YouTube changes its algorithm, it becomes such a fragile business structure. So, that was one of the reasons why I decided I needed to go into products.

HS I guess you can't really be known as the Queen of Shitty Robots where everything you make doesn't really do what it's meant to do, and then expect people to buy serious products from you and trust they'll work.

SG That's definitely one of the things when we launched The Every Day Calendar – I was wondering how are people going to be able to take this seriously? But, I think that's what's really nice, that my audience has been around long enough and they've seen that there's more to it than that – there's actually, ironically, a lot of work that goes into making Shitty Robots.

HS I remember the first time I saw your work was when you collaborated with Adam Savage to make an automated popcorn machine in 2016. It's a great video that really highlights how great collaborative work can be when two people focus on what they're really interested in to make a final product. And you've worked on other videos with creators such as Laura Kampf. Is there anyone else you'd like to work with?

SG I'm really interested in people who are kind of beyond the community that I'm currently in. It would be really fun to do stuff with musical artists; I'd love to collaborate with OK Go. Or venture beyond that and work with people who make art, and fashion designers. People who are outside the world where I'm creating. And there are people that I just love and would always want to work with, like Laura. She's the sweetest, most talented, down-to-earth and funny person. I really love working with her. I should really think of who's on my bucket list.

Something I've really missed during the pandemic is just getting to spend time with people who are excited about what they're doing, and having that excitement rub off on me. There's nothing more inspiring than someone being →

pumped about something, even if you don't understand what it is. In some ways, lockdown has been great for creating as I've had more time to loiter in the shop, but I definitely miss that input and just being able to talk to people.

HS And are there any projects you'd like to build that you just haven't gotten around to doing yet?

SG Honestly, I just want to build stuff for my house right now, which I know isn't the most interesting answer. I still have the CEO Bouncy Chair on my list – I want to make this kids' bouncy chair, the type where you're almost in some sort of plastic diaper. But I want it to look like a mahogany desk with a Rolodex and it's for grown-ups. And make some spoof commercial for it when it's marketed as an exercise device, but there's just some balding white guy in it. I think that's the only one that I'm still eager to build. Let me look at my notes...

[Simone proceeds to pull out her phone and list project ideas from the notes app. Should I tell you what they are or should I leave them as a surprise? With great power comes great responsibility!]

HS Those are definitely some interesting ideas...

[I'm very responsible].

HS Going back to your audience, you seem to have been somewhat spared a lot of the negativity people receive in comments, and online in general. Why do you think that is?

SG I'm just always so scared. Haha. I've been spared from the trolls and the hate, and I'm just terrified of ruining whatever equilibrium is happening right now. That's one of the reasons I post so seldomly. I was looking the other day and thought, 'Oh, it's been 45 days since I last posted on Instagram!', and I notice I keep getting DMs from people asking if I'm OK. I'm just always scared to overstep, or do something that would upset people, or cause me to fall from some sort of pedestal. I just never want to post something that doesn't work for other people, you know?

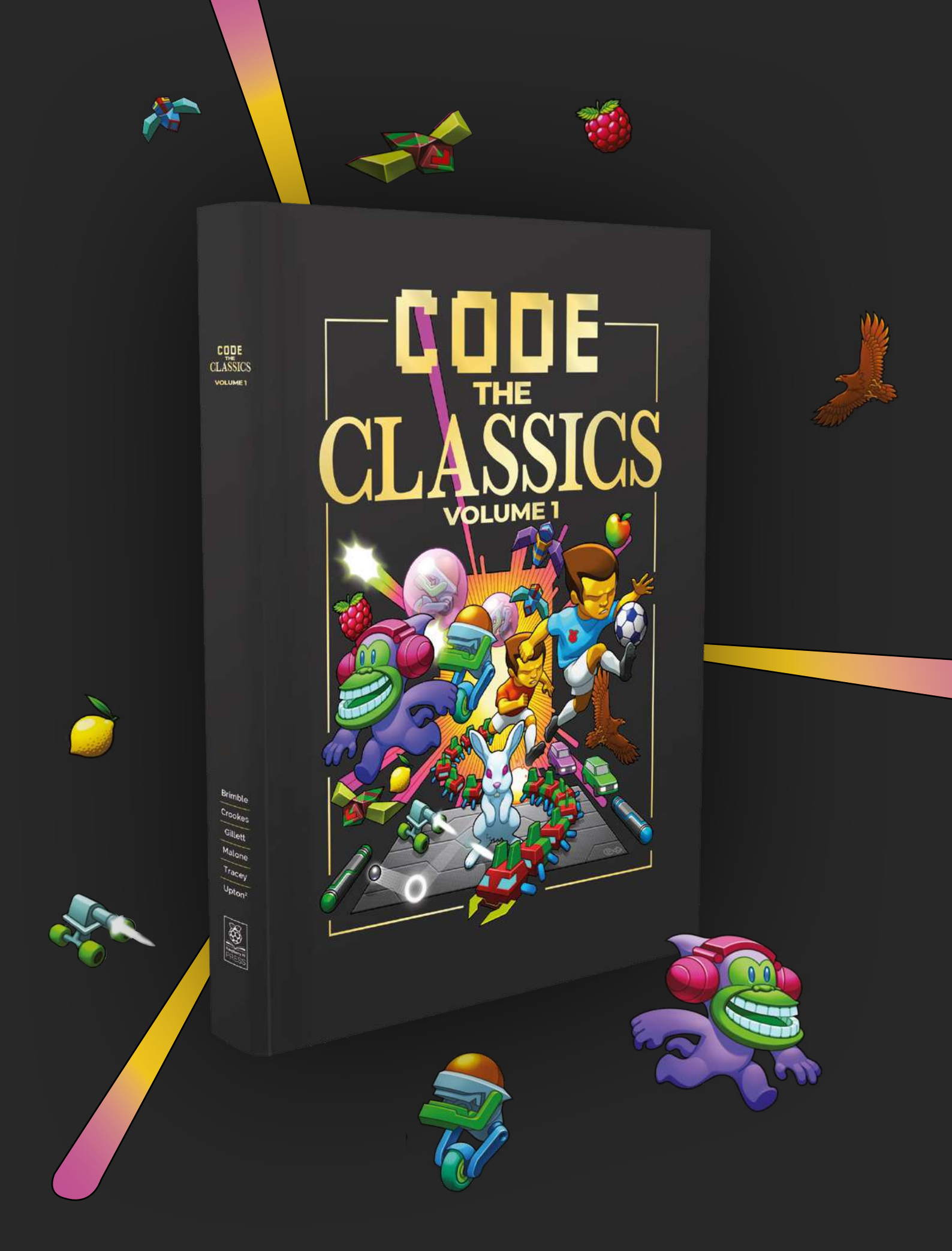
HS I get it. The comments section of YouTube alone can be an awful place sometimes. Speaking of YouTube, are there any other makers at the moment who are inspiring you?

SG I love 3x3 Custom. She's my happy place because she's at a level of making that I'm just not at. Her jig work is just wild, and the quality she puts out. And I love Nicole McLaughlin. She does these really fun and weird fashion contraptions, like shoes made out of tennis balls. She's very cool. She's a level of coolness that I aspire to and never expect to get to.

But, one of the most inspiring things for me is time. And I know that if I run out of ideas, it's because I'm overworked and I haven't had enough downtime and time to just loiter in the shop. I try to enforce this on Fridays, where me and my teammates just work on whatever project, and it doesn't have to be work-related. And some of my best ideas have come from that type of work, where I don't know what my end goal with this is, but I'm just going to tinker with it for a little bit.

You can follow Simone on Instagram (instagram.com/simonegiertz) for behind-the-scenes photos of her project, and subscribe to her YouTube channel (youtube.com/c/simonegiertz) for new content. Also, because why wouldn't you, you can follow Scraps on Instagram (instagram.com/commanderscraps) too!

Above
Scraps is first and foremost a dog, but he's also the brains behind Simone's Companion Chair


CODE
THE
CLASSICS
VOLUME 1
Brimble
Crookes
Gillett
Malone
Tracey
Upton²

IMPROVISER'S TOOLBOX

PALLETS

With timber hard to come by, Rosie Hattersley suggests four great ways with reclaimed wood

Rosie Hattersley

@RosieHattersley

Rosie Hattersley writes tech, craft, and life hacks and tweets **@RosieHattersley**.

With wood (and other DIY materials) hard to come by right now thanks to a combination of global supply issues, finding an alternative source of timber may allow you to complete projects that would otherwise be on standby. Typically found discarded on pavements after a delivery of large electrical goods or home improvement materials, pallets have been around for the past century. There are differing accounts of who first dreamed up the idea of pallets, but the first such US patent was granted in 1925 to one Howard T Hallowell for his Lift Truck Platform. Given the invention in the preceding two decades of various vehicles in the UK and US that closely resembled what we now know as the fork-lift truck, a standardised pallet size and design made sense.

As you'll discover during any mission to rescue and recycle such handy pieces of woodwork, pallets come in many varieties. Depending on what you intend to use them for, you should look out for a stringer pallet, which has two rows of slats, meaning a decent haul of wood to chop up and use, or block pallets, which are more structural, with widely spaced blocks underneath and slats only on the top. Use domestic pallets (which are unmarked) or ones labelled HT, DB, or EPAL, and avoid any marked MB, as these have been treated with methyl bromide – a poison.

There is also plenty of variation in terms of how easy a pallet may be to disassemble: some pallet-making masters suggest workarounds so you don't need to remove the most stubbornly embedded nails, while others propose homemade tools specifically for this purpose: **instructables.com/Pallet-Breaker**.

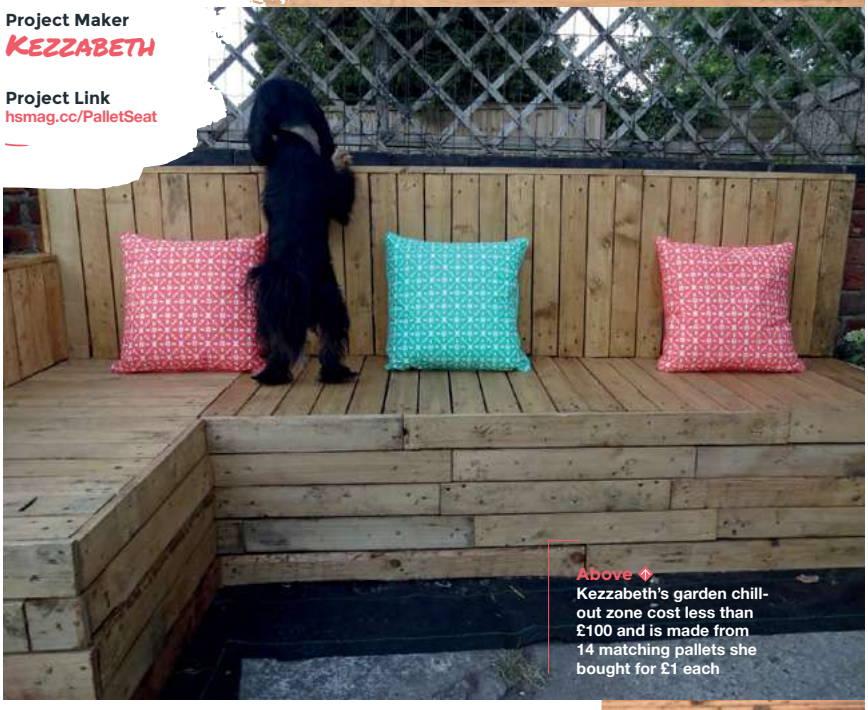
PALLET SEATING

Project Maker
KEZZABETH

Project Link
hsmag.cc/PalletSeat

Blogger Kezzabeth wasn't content with simply stacking and nailing together a couple of pallets as an outdoor seat. She deconstructed, and then assembled, 14 pallets into a sturdy corner sofa that's ideal for an outdoor gathering. Rather than take a chance on finding suitable discarded pallets, she bought a job lot of the same design for £1 each. Kerry set about removing the middle slats from her pallets, leaving a square box shape. The discarded wood from these pallets, and the timber from four pallets she fully disassembled, were set aside. Having treated the wood with decking preserver to weather-proof it, Kerry corralled her stacks of pallets into a rough corner sofa arrangement and nailed sets of three stacked pallets to each other. The sofa base was placed on thick weed-suppressant fabric. With a handy brick wall behind the sofa, it made sense to add a back-rest made from 100 cm lengths of exterior-treated timber.

Next, the lengths of wood that had been removed earlier were cut into even lengths, then nailed into place to form the seat and sides of the sofa. The back-rest came into its own here as it helped secure the planks in place. The whole thing was thoroughly sanded down, and leftover wood blocks used to create an armrest. Kerry suggests using sturdier wood blocks for the armrest, and adding extra nails to secure it since there's every chance people will lean heavily on it. Planks were added to the top edges to neaten everything, and cushions added for comfort. →



Above
Kezzabeth's garden chill-out zone cost less than £100 and is made from 14 matching pallets she bought for £1 each

"THE BACK-REST CAME INTO ITS OWN HERE AS IT HELPED SECURE THE PLANKS IN PLACE"

RAISED BEDS

Project Maker
JAMES PRIGIONI

Project Link
hsmag.cc/PalletBeds

Photos of pallet gardens abound online, but the basics begin with a well-planned and well-constructed raised bed. The Gardening Channel provides an excellent YouTube walkthrough. James advises ignoring the nailed sections and salvaging the wooden slats in between. This saves on time and effort, and results in a fairly even set of wooden lengths to use for the verticals of the raised bed. You'll need roughly 54 planks for a 4×8ft bed – the width of each pallet plank affects the amount required. Use a set-square to fine-tune the cut pieces to match. By using a drop saw set to 0 degrees, you can quickly work through the planks and apply the same setting. Use one of the remaining pallet frames as a brace to attach the planks to and to set the bed length. If some boards are wider, put these at the outer edges so that there's a strong board to attach to. Apply linseed oil to preserve the wood, give it ten minutes to sink in, then wipe off excess before brushing oil onto the next side. A special corner clip holds two sections of the pallet-board raised bed sides in place at right angles while you screw them to a vertical block. James advises pre-drilling before you attempt to attach screws. A top rail made from lengths of 2×4 is used to neaten and strengthen the structure. Homemade compost, a layer of rotten leaves, and one of mushroom compost are used for the basis of the bed, cutting down on the amount of bought-in compost needed to fill it ready for planting.

Right
Raised beds, ponds, and even hot tubs, all built from pallets, can transform a garden

PALLET WOOD GUITAR

Wood working expert JackmanWorks lovingly crafted his own guitar from a combination of specially selected pallet woods. A combination of maple, cherry, oak, poplar, mahogany, and pine pallet slats were first planed down using a thickness planer to ensure each slat was smooth and equal in depth. Jackman then cut each piece to a uniform two-inch width, and used a cross-cut sled to trim them to 23-inch lengths, before clamping and laminating the now barely recognisable pallet strips together. He used friend Tim's CNC machine to cut out a basic guitar blank shape with countersunk screw holes and a cavity where the neck would be attached.

The guitar body was laser-engraved and organic shapes crafted by hand using an angle grinder. Jackman then attached a premade guitar head unit to use as a guide for the fretboard and pickup placement. Holes were bored for the electronics, wires, and knobs, with slug tape hidden inside the guitar used to earth the electronics. The bridge was screwed into place and a strap attached. As "hipster millennials", it was a "requirement" that Jackman and Tim used repurposed vinyl records to cover the electronics. Finally, the guitar was given a 220 grit sanding and finished with five coats of tung oil.

Left
A selection of six different pallet woods were used as the basis of this expertly crafted guitar

Project Maker
JACKMANWORKS

Project Link
hsmag.cc/PalletGuitar

PALLET TIKI BAR

Project Maker
JODI HOUSE

Project Link
hsmag.cc/PalletBar

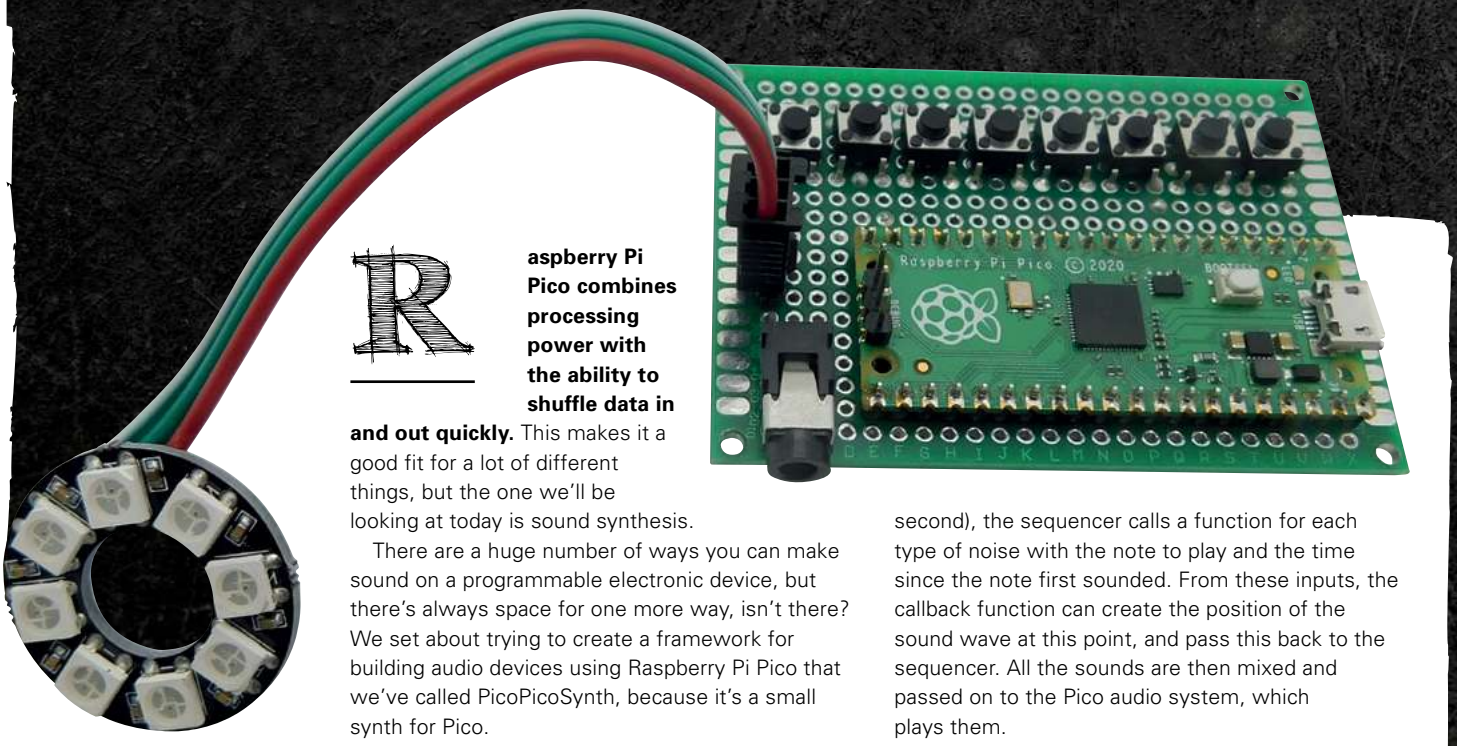
There are plenty of fairly straightforward outdoor bar designs that reclaim pallet wood; Jodi House's tiki bar has a little more panache, not least because it nestles among some tropical foliage. Jodi and her husband began by sketching out the bar's footprint, deciding two 4ft pallet widths would suffice to seat four at bar stools. Six 4×8ft posts, anchored with ground master post holders, give the bar structural integrity. Corner brackets and tie plates secure the pallets in place. A piece of 2×8 laid along the top of the weather-proofed pallets evens out the pallet-based front wall. In an ideal world, Jodi hoped to use an old surfboard for the bar top, but they couldn't find one they liked during the build. The roof is made from picket fences and the whole thing shaded by an existing palm tree. Once the bar was assembled, it was time to complete the beachside look with coats of Tropical Lagoon colour paint. ◻

FEATURE

IN THE WORKSHOP: PicoPicoSynth

By **Ben Everard**

An audio framework for Raspberry Pi Pico



Above
We're working on a DIY audio board to accompany this software. Hopefully, we'll be able to show it off in a future issue

Raspberry Pi Pico combines processing power with the ability to shuffle data in and out quickly. This makes it a good fit for a lot of different things, but the one we'll be looking at today is sound synthesis.

There are a huge number of ways you can make sound on a programmable electronic device, but there's always space for one more way, isn't there? We set about trying to create a framework for building audio devices using Raspberry Pi Pico that we've called PicoPicoSynth, because it's a small synth for Pico.

The program is powered by a sequencer. This is a structure that contains (among other things) a sequence of notes that it plays on a loop. Actually, it contains several sequences of notes – one for each type of noise you want it to play. Each sequence is a series of numbers. A `-1` tells the sequencer not to play that note; a `0` or higher means play the note. For every point in time (by default, 24,000 times a second), the sequencer calls a function for each type of noise with the note to play and the time since the note first sounded. From these inputs, the callback function can create the position of the sound wave at this point, and pass this back to the sequencer. All the sounds are then mixed and passed on to the Pico audio system, which plays them.

This setup lets us focus on the interesting bit (OK, the bit this author happens to find interesting – other people may disagree) of making music. That is playing with how manipulating waveforms affects the sound.

You can find the whole PicoPicoSynth project at **hsmag.cc/GitHubPPSynth**. While this project is ongoing, we've frozen the version used in this

article in release 0.1, which you can download from **hsmag.cc/PPSythV1**. Let's take a look at the **example_synth** file, which shows off some of the features.

You can create the sound values for PicoPicoSynth however you like, but we set it up with wavetables in mind. This means that you pre-calculate values for the sound waves. Doing this means you can do the computationally heavy work at the start, and not have to do it while running (when you have to keep data flowing fast enough that you can keep generating sound).

The wavetables for our example are loaded with:

```
        low_sine_0 = get_sinewave_table(50,
24000);
        low_sine_1 = get_sinewave_table(100,
24000);
        bongo_table = create_wavetable(9054);
        for (int i = 0; i < BONGOSAMPLES; i++) {
                bongo_table->samples[i] =
bongoSamples[i] * 256;
        }
```

The first two create sine waves of different frequencies. Since sine waves are useful, we've created a helper function to automatically generate the wavetable for a given frequency.

The third wavetable is loaded with some data that's included in a header file. We created it by loading a bongo WAV file into **hsmag.cc/WavetableEd**, which converts the WAV file into a C header file. We just have to scale it up from 8 bits to 16 by multiplying it by 256. We don't have a helper function to do the job here, so we have to load in the samples ourselves.

That's the data – the other thing we need are the callback functions that return the values we want to play. These take two parameters: the first is the number of samples since the note was started, and the second is the note that's played.

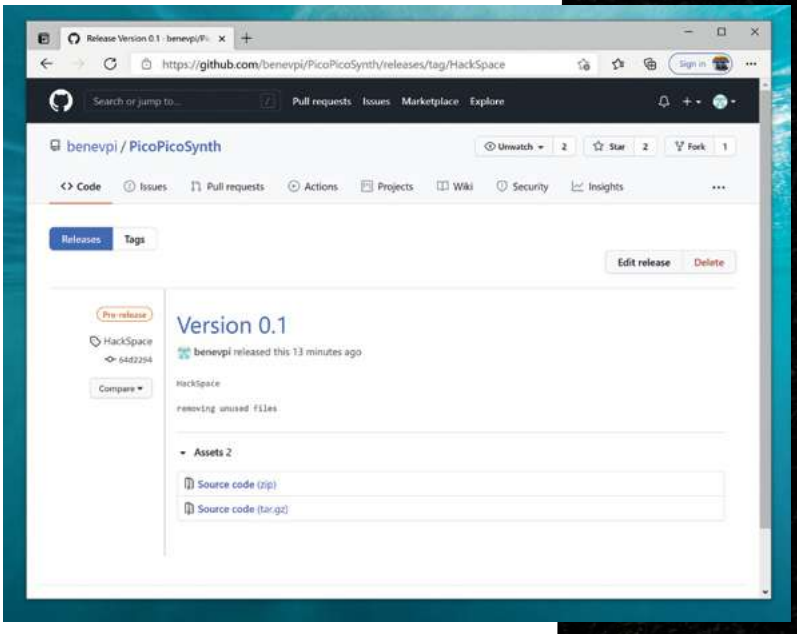
```
int16_t bongos(int posn, int note) {

        if (note == 0 ) {
                return no_envelope(bongo_table,
1, posn);
        }
        if (note == 1 ) {
                return no_envelope(bongo_table,
0.5, posn);
        }
        else {
                return 0;
        }
}

int16_t low_sine(int posn, int note) {
        if (note == 0 ) {
                return bitcrush(envelope(low_
sine_0, 1, posn, posn, 5000, 10000, 15000,
40000),32768,8);
        }
        if (note == 1 ) {
                return bitcrush(envelope(low_
sine_1, 1, posn, posn, 5000, 10000, 15000,
40000),32768,8);
        }
        else {
                return 0;
        }
}
```

Audio output

The Pico audio system can output sound as I2S, SPDIF, or even plain old PWM. With this latter style, you don't need any hardware other than a way of attaching your headphones (or line-audio-out jack) to the GPIO pins. At the time of writing, however, PicoPicoSynth only supported I2S because that's the hardware that this author's working with.

There are some examples of using the other code at **hsmag.cc/PicoPlayAudio**. If you want to get one of them working with PicoPicoSynth, it should just be a case of ensuring the includes are sorted out and creating an equivalent of init_audio_i2s (this is largely copied from the sine wave example in pico-playground anyway).

Pull requests are open, so if you do expand the library, let us know so we can share your enhancements with other users.



Below
You can grab the latest version of this code from GitHub

The note is 0 or higher – it corresponds to the number in the sequence, and you can use this however you like in your program. As you can see, both of our functions play sounds on notes 0 and 1.

The library includes a few functions to help you work with wavetables, the main two being **no_envelope** and **envelope**. The **no_envelope** function also takes a multiplier – it's 1 in the first instance and 0.5 in the second. This lets us speed up or slow down a sample, depending on what we want to play.

An **envelope** may be familiar to you if you've worked with synths before, and it's used to convert a constant tone into something that sounds a bit like an instrument being played. You supply four values – the attack, decay, sustain, and release times. During the attack phase, the volume ramps up. During the decay phase, it partially drops to a level that it holds during the sustain phase, and finally it drops to 0 during the release phase. This gives a much more natural sound than simply starting or stopping the sample.

The **envelope** function also has a multiplier, so we could use the same wavetable for both, but it's more accurate to generate a specific wavetable for each note if you've got the space to store it.

There are also a few sound effects in the synth library that you can apply – BitCrunch, for example. This compresses the sample bit depth down to give the sine wave a distorted sound.

These callbacks don't have to be sound. You could equally use them to co-ordinate a lighting effect, control physical hardware, or do almost anything else.

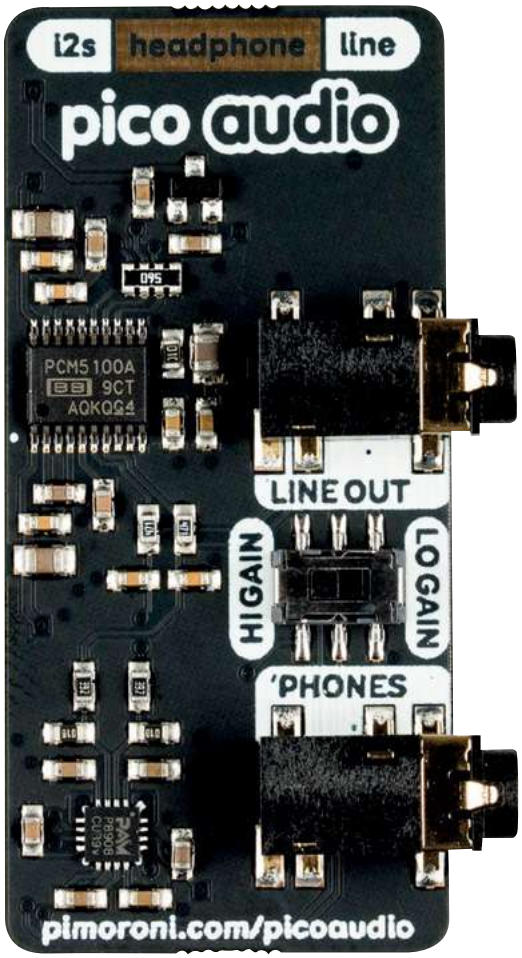
Now we've got the sounds set up, it's time to link them all together. This is done with the code below.

```
        int bongo_sequence[] = {1, 1, -1, -1, -1, 0, -1, -1};
        int low_sine_sequence[] = {-1, -1, 1, -1, -1, -1, 0, -1};

        struct sequencer main_sequencer;
        init_sequencer(&main_sequencer, BEATNUM, BEATFREQ);

        //add up to 32 different sequences here
        add_sequence(&main_sequencer, 0, bongo_sequence, bongos, 0.5);
        add_sequence(&main_sequencer, 1, low_sine_sequence, low_sine, 0.5);
```

Sequences are stored as **int** arrays that have to be the same length as the sequencer (stored in the **BEATNUM** macro). This can be any integer up to 32. The numbers in here can be anything you like, as they're just passed back to the callback functions defined above. The sole limitation being that only numbers 0 or greater are played. We also pass the **BEATFREQ** value which contains the number of samples per beat.

The final step in setting up the sound is to add up to 32 different sequences to your sequencer.

During the attack phase, the volume ramps up



Right
We used Pimoroni's Pico Audio Pack, but other forms of audio output should be possible

With everything set up, you can set the music playing with:

```
    while (true) {

        //do any processing you want here
        give_audio_buffer(ap, fill_next_
buffer(&main_sequencer, ap, SAMPLES_PER_BUFFER));
    }
```

Each time this loops, it calculates the next 256 (as stored in the **SAMPLES_PER_BUFFER** macro) and passes them to the audio system. You can do any other processing you like in the loop, provided it can run fast enough to not interrupt the sound playing.

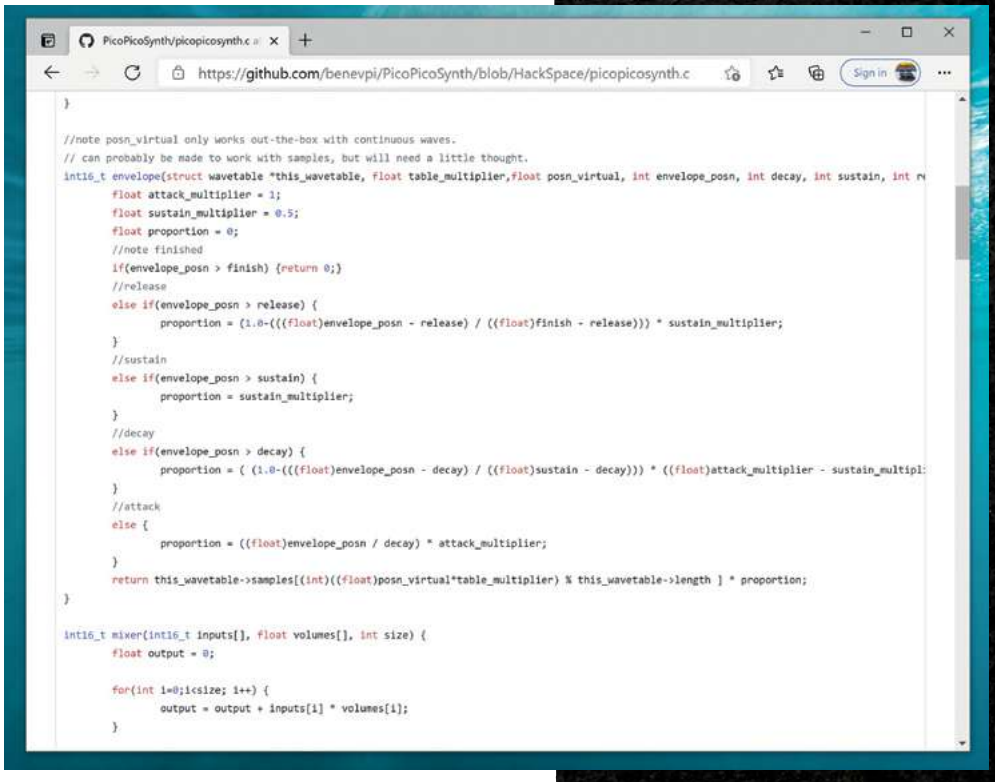
That's all there is to it. Set this code running on a Pico that's plugged into a Pimoroni Audio Pack (you should be able to make it work with other audio systems – see the 'Audio output' box, overleaf) and you'll hear some strange bumps and wobbles.

Of course, it's unlikely that you'll want to listen to exactly this strange combination of distorted sine waves and low bitrate bongos. You can take this base and build your own code on top of it. The callback functions can do anything you like, provided they run quickly enough and return a 16-bit integer. How you use this is up to you. □

Pico Extras

At the time of writing, the Pico audio system isn't part of the main SDK, but it's in a repository called Pico Extras (**hsmag.cc/PicoExtras**). This is a sort of staging repository, where bits of the SDK that aren't quite stable enough to make it into the main SDK live. In here, you'll also find scanline video (which is for outputting to DPI and DBI screens), SD card control, the sleep API, support for networking via LWIP, including a web server library and more.

While this isn't completely finalised, the audio section is quite stable, so it shouldn't change much. If there's an odd tweak here or there, we'll try to update PicoPicoSynth to keep up with this. We recommend taking a look at this repository, as there's quite a bit of useful stuff in there.



Left
The envelope function creates a donk, bong, bing, or other sound from a note with a constant volume

SUBSCRIBE TODAY

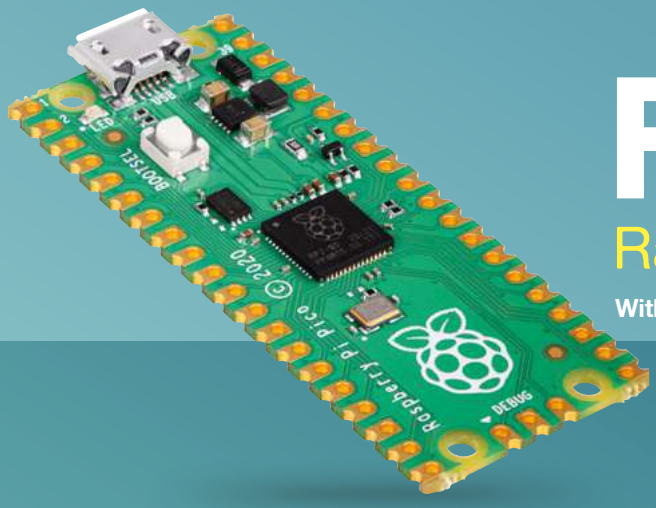
Get 12 issues of HackSpace magazine delivered to your door for just

£55 (UK)

£90 (USA)

£80 (EU)

£90 (Rest of World)



FREE!

Raspberry Pi Pico

With your first 12-month print subscription

This is a limited offer. Not included with renewals.
Offer subject to change or withdrawal at any time.

Subscribe online: hsmag.cc/subscribe

Subscription starts with next issue

HackSpace
TECHNOLOGY IN YOUR HANDS

FORGE

HACK | MAKE | BUILD | CREATE

Improve your skills, learn something new, or just have fun tinkering – we hope you enjoy these hand-picked projects

PG 76
CNC
We pit our budget CNC against aluminium

PG 82
ARCADE CONTROLS
Add retro controls to a Raspberry Pi arcade machine

PG 88
FREECAD
Design quickly with the Draft Workbench

PG 74
SCHOOL OF MAKING
Start your journey to craftsmanship with these essential skills

74 CircuitPython on MicroPython

PG 94
LASER FOCUSED

Build a bed height adjuster for your laser cutter

PG 98
WEB DEVICES
Talk to microcontrollers from your browser

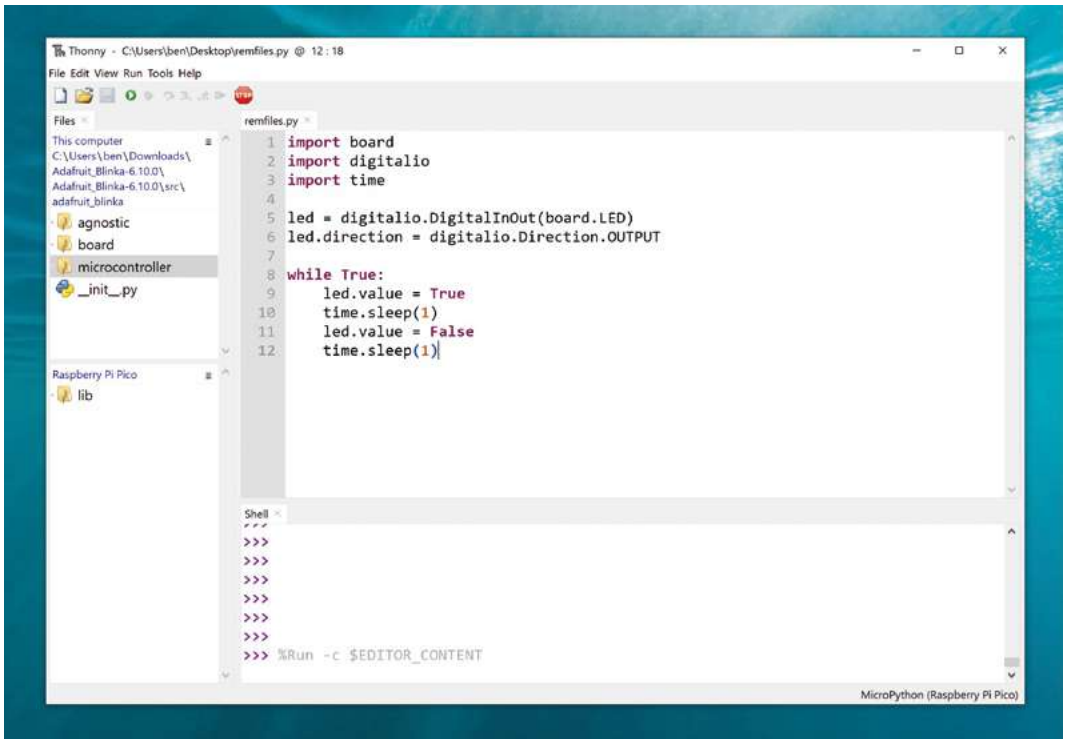
CircuitPython on MicroPython

Combine the libraries of CircuitPython with the hardware support of MicroPython

Ben Everard

@ben_everard

Ben's house is slowly being taken over by 3D printers. He plans to solve this by printing an extension, once he gets enough printers.



Above Using CircuitPython on MicroPython can give the best of both worlds

Raspberry Pi Pico has support for two different versions of Python: MicroPython, which is supported by Raspberry Pi, and CircuitPython, which is created by Adafruit. MicroPython unlocks a few more features of the underlying hardware, but CircuitPython has more libraries to support common maker hardware. Which should you use? How about MicroPython with a CircuitPython compatibility layer?

The CircuitPython compatibility layer – known as Blinka – has been around for a while and is best known for providing access to CircuitPython libraries for machines that run full-blown Python, such as Raspberry Pi computers. However, it originally started as a MicroPython compatibility layer, and it can still be used for this.

Let's take a look at how to get started with Blinka using Pico. We'll be using Thonny, so make sure you have an up-to-date version of this installed on your computer.

To start with, you'll need a Pico flashed with the latest version of the MicroPython firmware. First, unplug your Pico if it's already plugged in, then press the BOOTSEL button and hold it down while you plug Pico back into your computer. Once it's plugged in, you can release the button.

Open Thonny and go to Run > Select Interpreter. Select 'MicroPython (Raspberry Pi Pico)' in the drop-down and click 'Install or update firmware' in the

bottom right-hand corner. Your device's information should appear at the bottom of the pop-up window. If it doesn't, unplug it and plug it back in (with the BOOTSEL button held down), or try using a different USB cable.

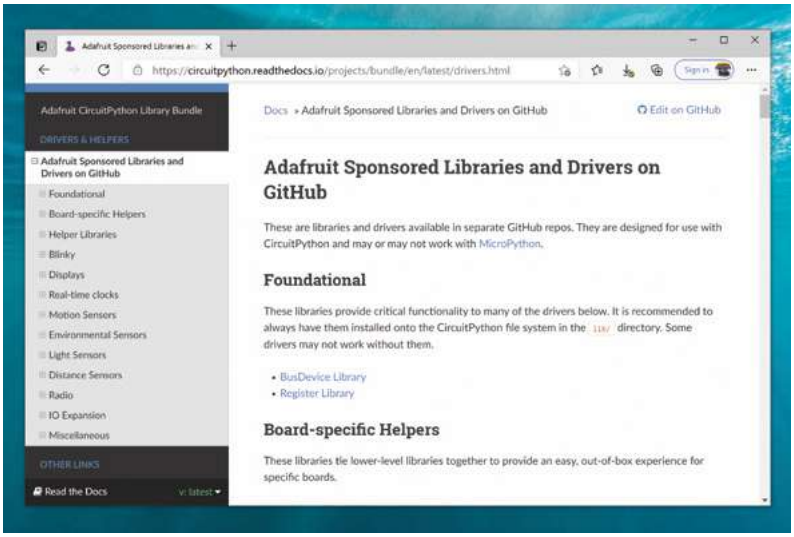
You can now close the pop-up and click OK in the Select Interpreter window to go back to Thonny. The Shell pane at the bottom of the window should now connect to the MicroPython interpreter running on Pico, and you should see '>>>'.

That's the MicroPython part sorted – now let's look at getting CircuitPython libraries up and running. There are two things we need: Blinka and Adafruit-PlatformDetect. You can download both of these from their GitHub repositories: **hsmag.cc/AdafruitBlinka** and **hsmag.cc/PlatformDetect**, respectively. You'll need to upload both of these.

In Thonny, go to the View menu and make sure Files is selected. This will open a Files pane on the left-hand side with two boxes: one for your computer and the other one for Pico. The Pico box will probably be empty, so you can right-click in there and select New Directory and call it **lib**. Then click on the newly created **lib** directory to enter it.

In the computer box, navigate to where you just unzipped the two zip files. You should have a folder called something like **Adafruit_Python_PlatformDetect-3.13.3**. Inside that, there should be a folder called **adafruit_platformdetect**. It's this inner folder you need to upload, so right-click on it and select 'Upload to /lib'. If this option is missing from the Context menu, make sure that you entered the **lib** folder in the Pico box.

The Blinka module contains a lot of bits that aren't needed for Pico, so it's best to remove these before uploading. In the **src** subfolder, go into **adafruit_blinka** > **board** and delete everything that isn't **raspberrypi** or **__init__.py**. In the **adafruit_blinka** > **microcontroller** directory, delete everything that isn't **rp2040** or **__init__.py**. Once this is done, upload everything from the **src** to the **lib** folder.



Above The CircuitPython libraries have excellent documentation

You can check that everything's worked by running the following code:

```
import board
import digitalio
import time

led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT

while True:
    led.value = True
    time.sleep(1)
    led.value = False
    time.sleep(1)
```

If you're familiar with CircuitPython and MicroPython, you'll see that the above is CircuitPython. The three modules it imports are all CircuitPython (though time is often aliased from `utime` on MicroPython). This particular program is just a test – you could equally well write it in MicroPython. However, the point is that you now have access to CircuitPython libraries. Not all will work, but those that use I2S and SPI should, which means you can use the huge range of sensors and actuators that communicate over these buses.

You can download the libraries for CircuitPython from **hsmag.cc/LibBundles**. You'll need the **adafruit-circuitpython-bundle-py** version, as MPY files for CircuitPython won't work with MicroPython.

Unzip this download and you can copy whatever libraries you need over to the **lib** folder on Pico. Once there, you can import them and use them as with any other module.

THE EASY OPTION

We've taken a look at how to set everything up on a blank copy of MicroPython. Of course, you don't have to start with a blank copy of MicroPython. The good folks at Pimoroni have created a UF2 file that you can drag and drop straight onto your Pico that includes Blinka and Adafruit-PlatformDetect (as well as some other bits and pieces).

This gives you the advantage of skipping the setup stage of this tutorial; however, it does have a couple of disadvantages. It's significantly larger than the plain version, so you'll have less free space on your Pico. Also, it has a few differences from plain MicroPython, including a slightly tweaked I2C interface. For simple projects, though, neither of these should present significant problems.

TUTORIAL

Advanced approaches with the CNC 3018 Pro

In this third part of the series, we look at some more advanced uses of the CNC 3018 Pro, 3D toolpaths, engraving, the laser module, and cutting aluminium

Jo Hinchliffe

@concreted0g

Jo Hinchliffe is a constant tinkerer and is passionate about all things DIY. He loves designing and scratch-building both model and high-power rockets, and releases the designs and components as open-source. He also has a shed full of lathes and milling machines and CNC kit!

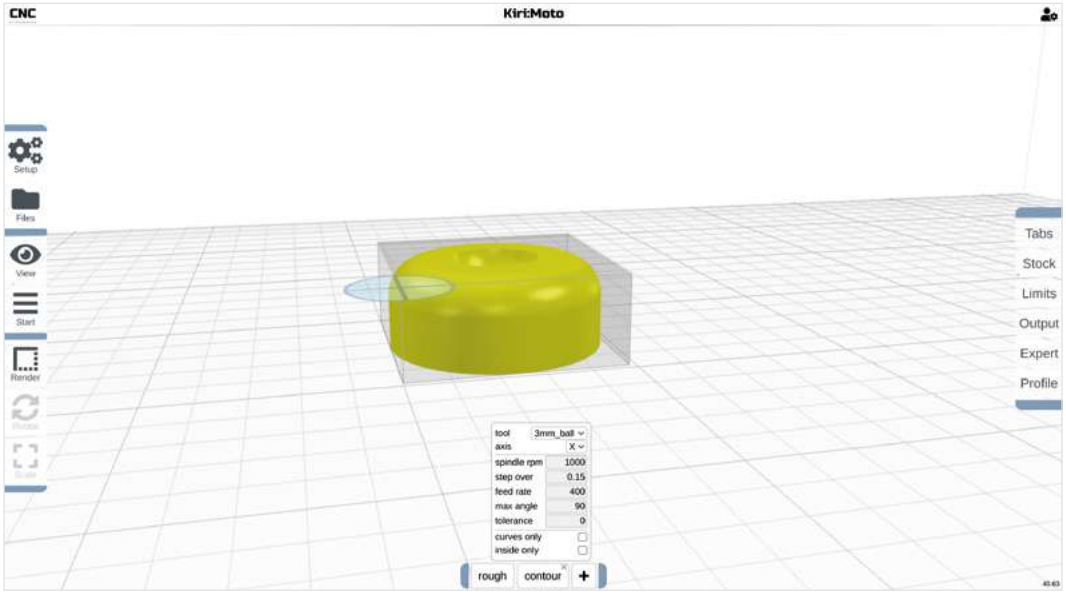


Figure 1 Kiri:Moto is an excellent piece of open-source software capable of creating G-code files for the CNC 3018 Pro and much, much more

YOU'LL NEED

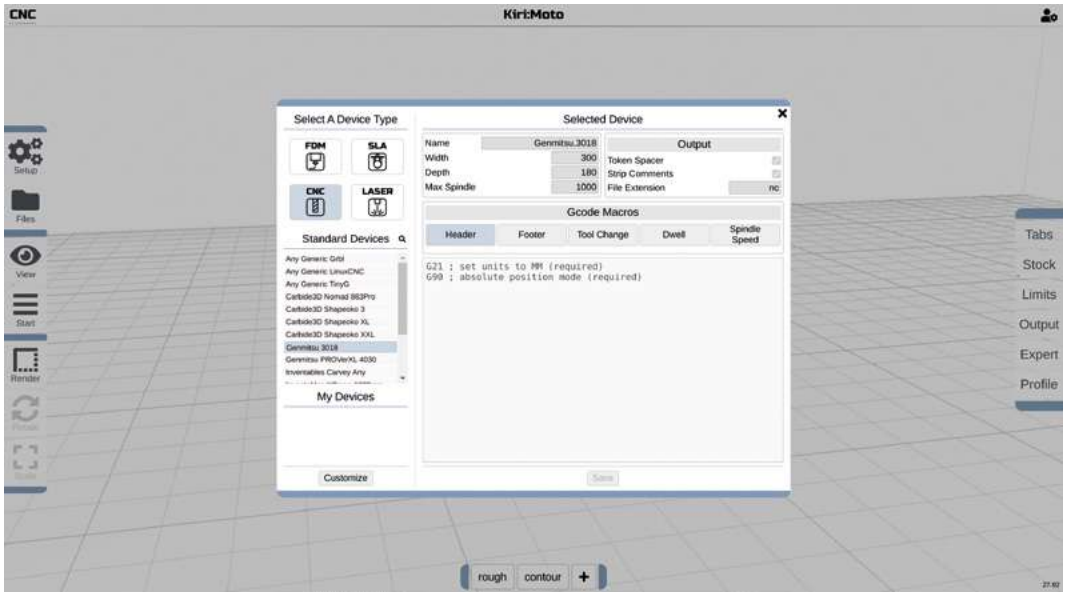
- **A CNC 3018 Pro kit**

Having performed some cuts in a variety of materials in the last article, we wanted to look at slightly more advanced CNC work this month using our CNC 3018 Pro. In this article, we will try our hand at true 3D CNC milling – creating a shape that's carved from a solid piece of material, rather than cut out of a flat sheet. We are also going to play with the laser module and share some ideas about engraving into materials. And we'll also ask the question that everyone asks about the CNC 3018 and other small CNCs – will it cut aluminium?

Last month we used KrabzCAM for our 2D milling experiments, cutting from flat sheets. To explore 3D milling, we are going to use another open-source package called Kiri:Moto (**Figure 1**). Kiri:Moto is an amazing set of web-based tools that can be used for all manner of CAM work. It's not limited to CNC toolpaths as it's a capable slicer for 3D printing and more. Kiri:Moto runs in the browser, but your projects and files are held locally and not uploaded to the cloud. Rather brilliantly, Kiri:Moto includes lots of machine profiles, including one for the CNC 3018 Pro, which means that a lot of the setup work is done for us.

First, open up Kiri:Moto by heading to **grid.space/kiri/**. Kiri:Moto launches in FDM mode, which you can use if you want to slice objects for 3D printing. You'll see the letters FDM in the upper left-hand corner; click on that and in the pop-up dialog, change this to CNC to enable the CAM section. Before you close that box, scroll down the standard devices list, then select the profile called 'Genmitsu 3018'. We don't need to make any further changes to this page, but you'll notice now that the machine dimensions and some start and stop messages and other bits and pieces are set up correctly for our machine (**Figure 2**).

We are going to cut an example project that we will load into Kiri:Moto from an STL file. We've drawn this object in FreeCAD, and it includes numerous curved



QUICK TIP

Make sure you click the 'metric' checkbox as, by default, all the measurements in the tool dialog are in inches unless this is ticked!

parts – while it isn't anything in particular, it serves as a great test object. To cut this object from some material, we will perform some different types of procedures – this is similar to last month where we performed both profile- and pocket-type cuts. The two procedures we will perform will be a 'roughing' cut, followed by a 'contour' cut. The roughing cut will remove material from the stock to clear the area for the contour cut to bring the object to the final shape. We'll use a 4mm end mill for the roughing cut, and a 3mm ball end mill for the finishing contour cut (**Figure 3**).

By default Kiri:Moto contains a few cutting tools, but we will create our own tool profiles for this job. Clicking 'Setup' on the left-hand side of the screen, we can then select 'Tools', followed by the '+' button to add a new tool. First, we added a 3mm ball end mill cutter, selecting Ball from the cutter type drop-down and inputting the dimensions of our tool. We then repeated this for our 4mm end mill tool.

When you open Kiri:Moto for the first time, a default cubed item is placed on the stage; we left-clicked on this item and pressed the **DELETE** key to remove it. Replacing it with our project STL, we clicked the 'files' icon on the left-hand side and then 'import'.

Having imported the item, we noted that the stock defaults to the size of the item. For our project, and for others, it's useful to be able to adjust this. On the right-hand side of the screen, clicking the 'stock' tab, we can see some dialogs relating to stock sizes and behaviour. The 'width', 'depth', and 'height' input boxes can be used to add extra amounts of stock to the default cuboid of stock material Kiri:Moto places around your imported object. For example, we added 5mm to both the width and the depth of the stock, and we increased the height by 2.5mm as our object is 10mm high, but our stock MDF is 12.5mm thick (**Figure 4** overleaf). When we added the extra height, our project file was raised so that the surface of the object was at the top of the stock. Sometimes this might be preferred, but in this case we want to place the object at the bottom of the stock to cut the workpiece out fully. To rectify this, we clicked the 'Limits' tab, and then in the 'Z Anchor' drop-down menu we set it to 'bottom'. Before we create toolpaths, we want to add some tabs to our object so that the work is not flung out of the machine when completed. Clicking 'tabs' on the right-hand toolbar, we set the width to 5mm and the depth and height to 3mm. Then in the tabs dialog, click the '+' sign in the lower left-hand corner. Clicking the '+' sign sets you into tab editing mode, and you can hover over the lower parts of your object and see a tab appear in preview – move the tab to the position you require and left-click to place it. You can repeat this to add →

Figure 2
Kiri:Moto supports the CNC 3018 Pro with a decent preset for the machine

Figure 3
Ball end cutters are very useful when cutting 3D shapes on a CNC

QUICK TIP

Kiri:Moto keeps all your settings between projects locally, so your custom tools are available for future projects. None of your project data is stored in the cloud.

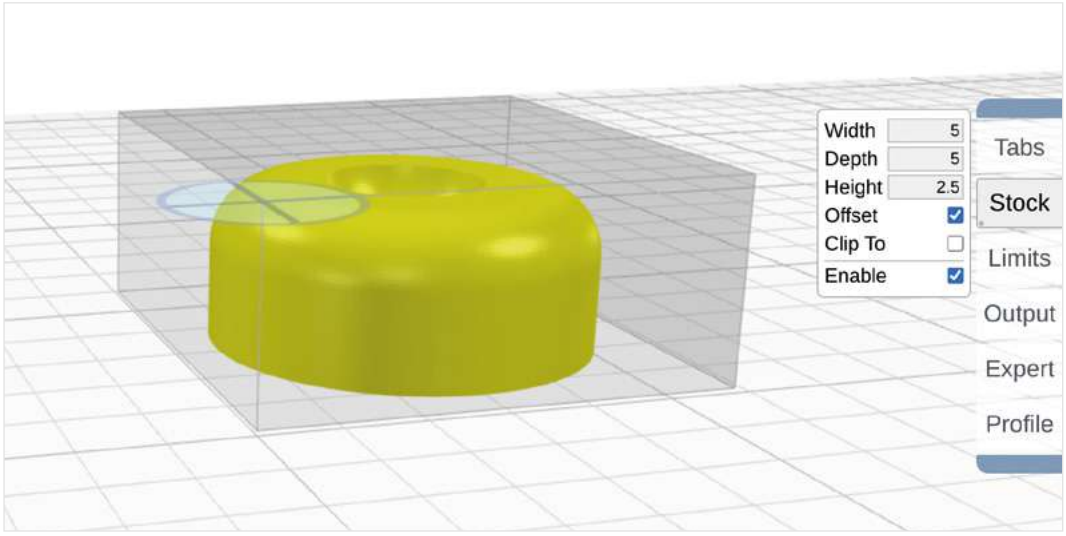


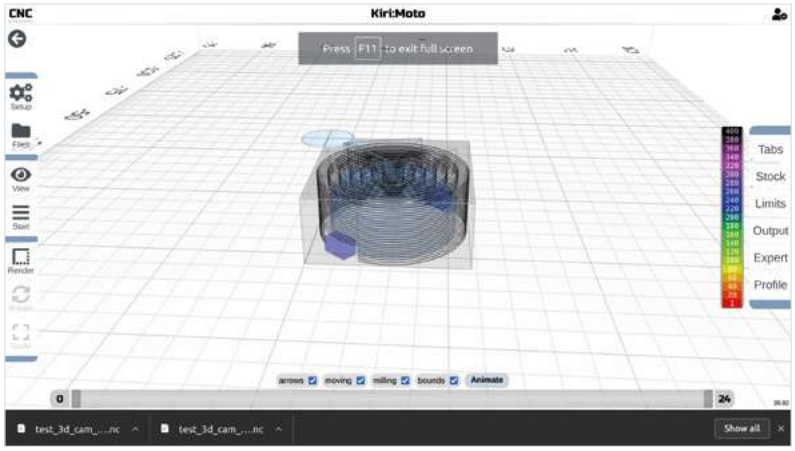
Figure 4
Our curvy test item imported to the Kiri:Moto stage

Figure 5
Tabs have been added to secure the workpiece, and the roughing operation has been created

as many tabs as you like. We added two tabs on opposite sides of the object and then hit the **ESC** key to drop out of tab editing mode.

SOMETHING FROM ROUGHING

To create our roughing operation, we clicked the '+' sign at the bottom of the screen and selected 'roughing'. We selected the 4 mm end mill we added earlier from the tool drop-down menu. We left the spindle speed-rate at 1000 and set both the stepdown and the stepover to 0.5, but note that those mean different values as the 'stepdown' is calculated in the workspace units, mm, whereas the 'stepover' is calculated as a fraction of the tool diameter. So a stepover of 0.5 on a 4 mm tool means that each clearing path will overlap by 2 mm. We were quite conservative with our feed rate of 400 mm/min, and again we set the plunge rate low to minimise Z axis deflection and went for 200 mm/min. In the 'leave stock' option, we added a small offset of 0.1mm. This means that our roughing cut will – even on the surfaces where this square end mill could



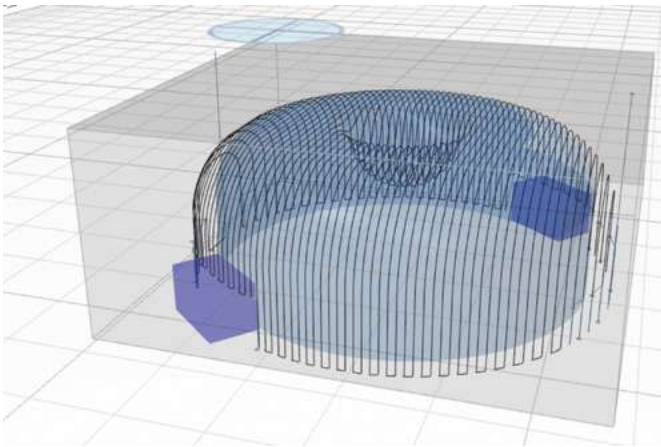


Figure 6
The contour path set up to finish our roughed out workpiece to size and shape

cut the project geometry to actual finished size – leave 0.1 mm of stock which we will remove in our contouring operation. Finally, we checked the clear voids, clear faces, and clear top checkboxes, leaving the 'Inside only' checkbox unticked. With everything set up, click the 'Start' icon on the left-hand toolbar and then select 'Preview'; Kiri:Moto will create the toolpath and display it (**Figure 5**). Once you are happy with your toolpath, you can click 'Start' and then 'Export' to download a G-code file for this operation.

We deleted the roughing operation (click the 'X' in the operation tab) and then set about creating our finishing contour toolpath. Click the '+' icon to add an operation and pick contour. In the dialog, we selected the 3 mm ball end mill tool we had set up and left the axis set to 'X'. In this project, as the object is symmetrical, there is no difference in terms of running the contours in the X axis or the Y axis;

for other projects, however, you can experiment with this. We left the spindle speed at 1000 and set the stepover to 0.15, with the stepover being again a fraction of the tool diameter.

A 0.15 stepover with a 3 mm diameter tool puts the contour toolpaths on 0.45 mm spacings and should give a reasonably detailed surface finish. We set the feed rate to 400 mm/min and set the max angle to 90 as the CNC 3018 Pro can't tilt the spindle. Leaving the tolerance at the default 0, we then previewed and, in turn, exported the G-code for this operation (**Figure 6**).

We set up the machine with the 12.5 mm MDF clamped on top of some waste board and inserted a 4 mm collet and the 4 mm end mill. We jogged the spindle to a suitable position, zeroed the Z axis, and then sent the roughing operation.

At the end of the operation, we need to move the spindle back to the zero position so that we precisely run the next job from the same origin point. Unlike other G-code-sending applications, there isn't a button for this by default in Candle. However, we can easily set up the 'Safe position' button to return us to our zero point. Go to settings, and in the dialog box under the heading 'Control', you'll find an input box for 'safe position commands'. Into this box, type 'G0 X0 Y0 Z0 and then click OK. Now, whenever you press the safe position button (it looks like a running person icon), the tool will return to the zero position. You do need to consider that the tool head is going to go in a straight line between its current position and the zero point. If, for some reason, you have an obstacle or your tool tip is still inside the workpiece, then you need to jog it to a safer position before doing this!

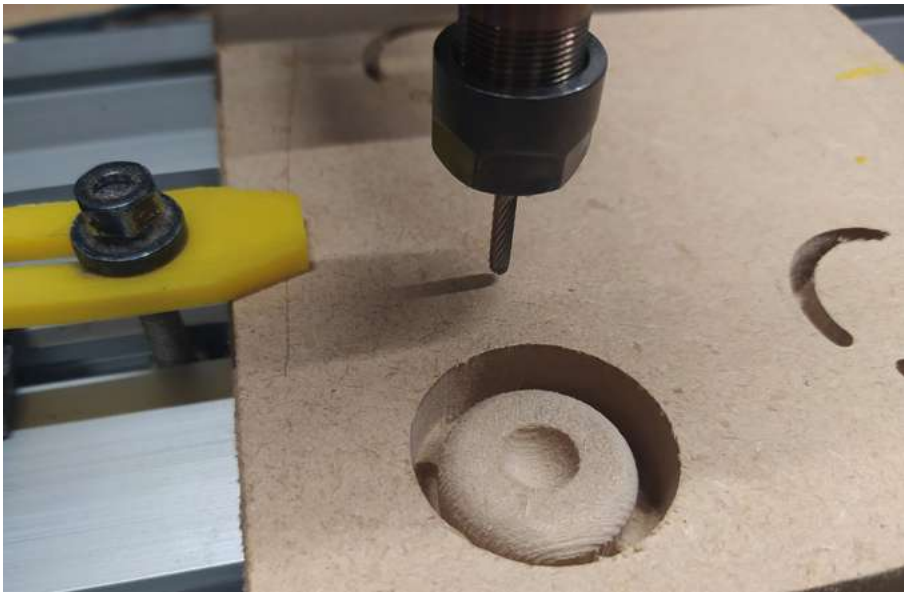
Once back in the zero position, we raised the spindle using the jog buttons, replaced the tool with the 3 mm ball end mill and then re-zeroed it down onto the work. Once ready, we ran the contour operation. It's really pleasing to see your 3D part emerge from the roughed out blank (**Figure 7**).

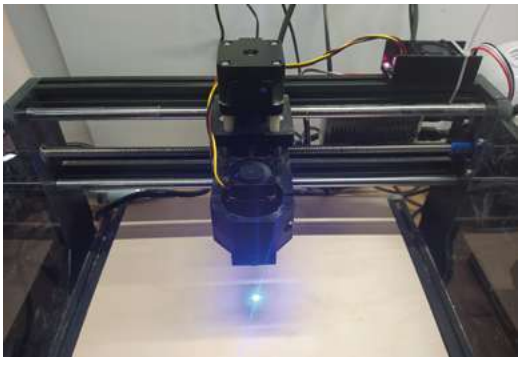
LASER FOCUS

We also wanted to test the laser module and try laser engraving using the supplied software. We loosened the spindle mount and then removed the spindle and unclipped the spindle cable from the control board before swapping the laser module. We loosely cable-tied the laser driver module to the X axis frame. Unlike the Candle software, the LaserGRBL software requires installation; however, a simple double-click to install and we were up and running. LaserGRBL is not only a controller, but it also acts as an image converter, importing and converting a range of image file formats ready to be laser engraved.

Using powerful lasers has a number of risks. There's the obvious risk to eyesight. Your laser module should have come with safety glasses and these should be for the same wavelength as your laser (not all laser safety glasses are suitable for all lasers). All people in the vicinity should be aware of the risks here and wear appropriate safety equipment. The laser will burn away the material it's etching, and this will put out a lot of smoke and fumes which are not healthy to inhale – make sure you properly understand the risk here and take appropriate mitigating steps. There's also the risk of setting your work piece on fire. Never leave a laser running unattended, and be prepared with the right equipment to put out a small fire.

We will need to focus the laser – our laser has a manual focus ring that adjusts the lens. It's also useful to be able to move the Z axis to get the laser module in a position that it's easy to focus, which may differ with material thickness. However, in LaserGRBL, the default settings have no jog controls for the Z axis. To enable the Z axis control, go to GRBL > Settings, →



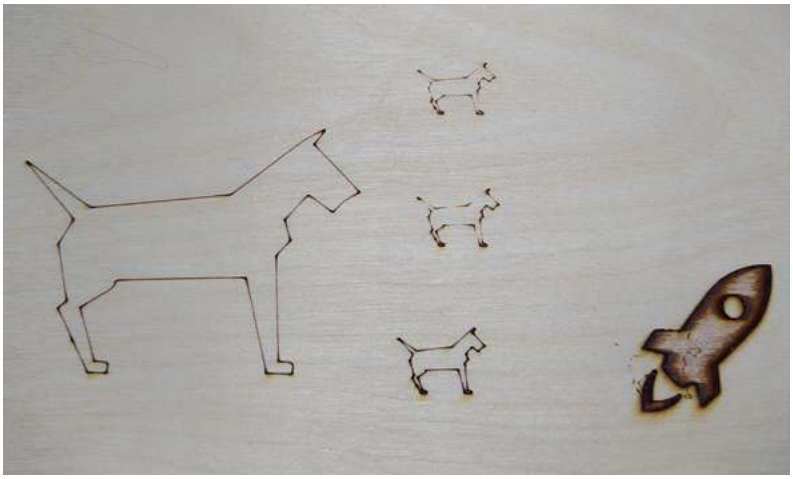


QUICK TIP

The laser module is dangerous. Ensure that you only ever view the laser whilst wearing laser glasses, and make sure no one, including pets, can enter the work area during operation.

Figure 7
Our workpiece is completed with the contour operation and still held in the stock by the small cut-away tabs

Figure 8
Attaching the laser module and manually focusing the beam



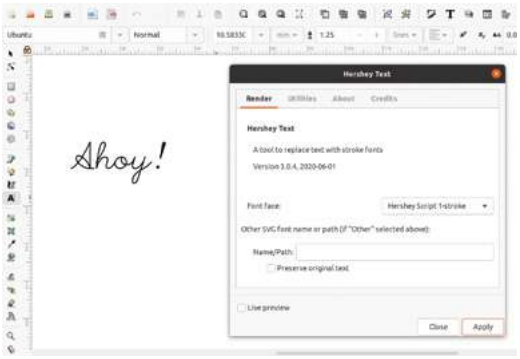


Figure 9
A collection of small tests using the laser module

Figure 11
Using Inkscape's Hershey Text extension to create single-line text suitable for engraving

Figure 10
The options and settings for importing images are pretty intuitive and straightforward in LaserGRBL

select the 'Jog Control' tab and tick the box marked 'Show Z up/down control' – you should now see a Z axis position control appear.

We wanted to test engrave into a piece of 3.125 mm birch plywood, so we jogged the Z axis close to the lower limit. In LaserGRBL, click to connect to the machine and place the workpiece on the bed. There's no cutting force when laser engraving, so it doesn't need to be particularly clamped to the workbench, but it's a good idea to use a little tape to stop it moving with machine vibrations.

> Once the laser is on, adjust the focus ring to create a **small spot with the laser on the material**

To focus the laser, put on the safety glasses and type 'M3 S40' into the command line and click 'Enter'. This turns on the laser with a power of '40' in a range of 0 to 255, which means it's not powerful enough to burn your fingers or the material (**Figure 8**). If the laser doesn't light, it is probably that the button on the laser module driver board needs pressing to activate the unit. Once the laser is on, adjust the focus ring to create a small spot with the laser on the material. Once ready, you can send an M3 S0 command to turn the module off.

To test the laser, we imported several image files, including SVG and PNG images – LaserGRBL is friendly in that it prompts you through converting the image. For an SVG with some single line vectors, like the dog outline in **Figure 9**, it's simply a case of setting the power and the 'boundary' speed (**Figure 10**). We experimented with various settings and, for our machine, settled on around 600 mm/min at full power.

It's fair to say that the laser works, but we wouldn't recommend this as the primary reason for you to buy the CNC 3018 Pro – with no air assist and no means of venting or cleaning the smoke built-in, you don't get the best-quality laser images.

Often the CNC 3018 Pro is bought with machine engraving in mind – our kit came supplied with some 20-degree engraving cutters. We were keen to try out some simple engraving to see how well the machine performed. As a first test, we decided to engrave some small and simple text into some fibreglass sheeting. Often when engraving text, it's useful to use 'Hershey' text. Hershey fonts use a single stroke to create each letter or symbol. Inkscape has an excellent Hershey Text extension. In Inkscape, input some text in any font and then click 'Extensions > Text > Hershey Text'. In the resulting dialog, we selected 'Hershey script 1-stroke' and then clicked Apply to get our word 'Ahoy!' converted (**Figure 11**).

Saving that file as an SVG meant we could open it in KrabzCAM – which we covered in last month's article – and create a G-code file to cut. You can see the settings we used for feed and speeds in

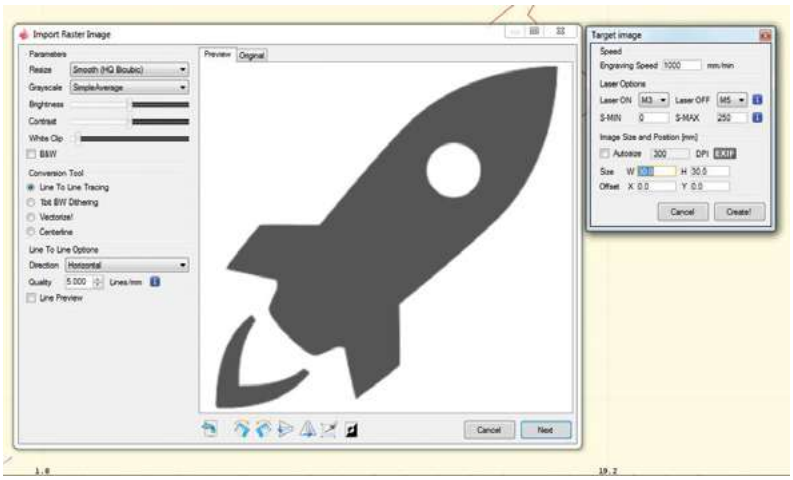


Figure 12. We set the spindle speed to full at 1000 by manually editing the G-code file, as we've done with previous KrabzCAM files.

The fine-pointed engraving cutters are very sharp, so do handle with care. The tips of these cutters are also very delicate and easy to break, they can be difficult to zero or touch on, and we've even broken the tip when using a feeler gauge to zero the tool! Next month, we will look at a hardware solution to make touching on much easier, but for this month we have a hacky cheat to set up these tools! Insert the tool into the collet holder, but only hand-tighten the collet enough at first to allow the tool to move inside it. Move the spindle into a start position and gently move the tool down the last few millimetres until you feel it touch the surface of the work, then tighten up the collet securely. This isn't the best approach, but it makes sure that the tip is zeroed and doesn't risk breaking. Once zeroed, we sent the job, ensuring we had adequate vacuum clearing the fine fibreglass chips – we also wore our PPE face mask.

With a depth of cut of 0.15mm, we got a quite pleasing engrave – to show it off a little, we carefully used a fine-tipped permanent ink pen to backfill the engraved section (**Figure 13**). In a future part of this series we will look at engraving further, exploring some software enabling PCB milling.

To conclude this month's article, it's fair to say we have found the CNC 3018 Pro a little machine that, out of the box, manages to punch above its weight, is capable of creating accurate parts, and is excellent to learn on. Next month, we are planning to do some small modifications and upgrades to this machine to make it even more useful. ◻

Figure 12
Setting up the engraving operation in KrabzCAM

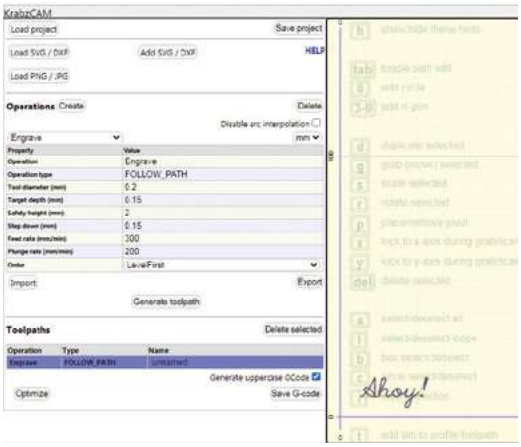


Figure 13
Our completed small engraving backfilled with some black ink

CUT ALUMINIUM

One thing that always seems to be asked of any CNC machine is if it will cut/machine aluminium. We were keen to give it a go, but we were not particularly hopeful because even our other much more rigid small CNC router is a bit underpowered for aluminium work. For our test, we decided to try a small pill-shaped pocket cut into some scrap 7075 aluminium plate. On less rigid machines, the first thing is to totally reduce your depth of cut, and with the CNC 3018 Pro having some flex in the Z axis, we decided to try 0.1 mm per pass. Using a two-flute 3mm carbide end mill, we set the stepover to be 25% of the tool diameter, which is 0.75mm. This means that for the first cut of any pocket layer, the tool removes 3mm width at 0.1 mm depth and then continues to cut the inner paths at 0.75mm width and 0.1 mm depth. The second thing to consider is to really slow down the feed rate; we went with 150mm/min and a 100mm/min plunge rate. We set up the path in KrabzCAM and then edited the G-code, setting the spindle speed to 1000.

When machining aluminium, the aluminium chips can stick to the tool, so using a lubricant will help make the cut successful. We use a proper cutting and taping fluid, but a squirt of WD-40 would do just as well. Running the job, the machine didn't seem stressed, so we probably could raise the feeds and cut depths a little. It did cut a reasonable pocket, not the highest quality of surface finish, but certainly accurate. It would have been better to cut the pocket a tiny amount undersize and then do an extra finishing cut to clean up the outer edges, but it did do the job, eventually!

Part 03

Build an arcade machine: Command and control

We've assembled our cabinet. Now it's time to put Raspberry Pi to work with the Recalbox arcade OS

MAKER

K.G. Orphanides

K.G. is a writer, maker of odd games, and software preservation enthusiast. Their family fully supports the idea of an arcade machine in the living room.

@KGOrphanides

With our arcade cabinet built, it's finally time to get emulating with Raspberry Pi. We're using the Recalbox emulation distro for this project, which has excellent GPIO arcade controller support, a slick front end, and a handy web interface to help you configure and manage it.

The RetroPie distro is a popular choice for arcade machines, and adds Steam Link support, but requires manual installation and pull-up switch reconfiguration to get GPIO arcade controls working.

01 Wire up your controls

Last month, when we added buttons to our cabinet, we recommended attaching the spade-to-DuPont cables that will connect to Raspberry Pi's GPIO before inserting the buttons. If you didn't, it's time to open the back of your cab, grab a torch, and get in there to fit them.

Connect a spade-to-DuPont cable to each button and connect a shared ground cable to each of the left and right button banks. Where you have longer stretches of buttons – for example between the central hot button connected to player one's rig and the player one start button – it's a good idea to skip a connector on the ground chain to give yourself some extra cable to play with.

Plug the 5-pin cable into the joystick. Looking at our Sanwa stick from below, the bottom-most pin, which connects to the black cable strand on standard-coloured 5-pin wiring harness, is ground.

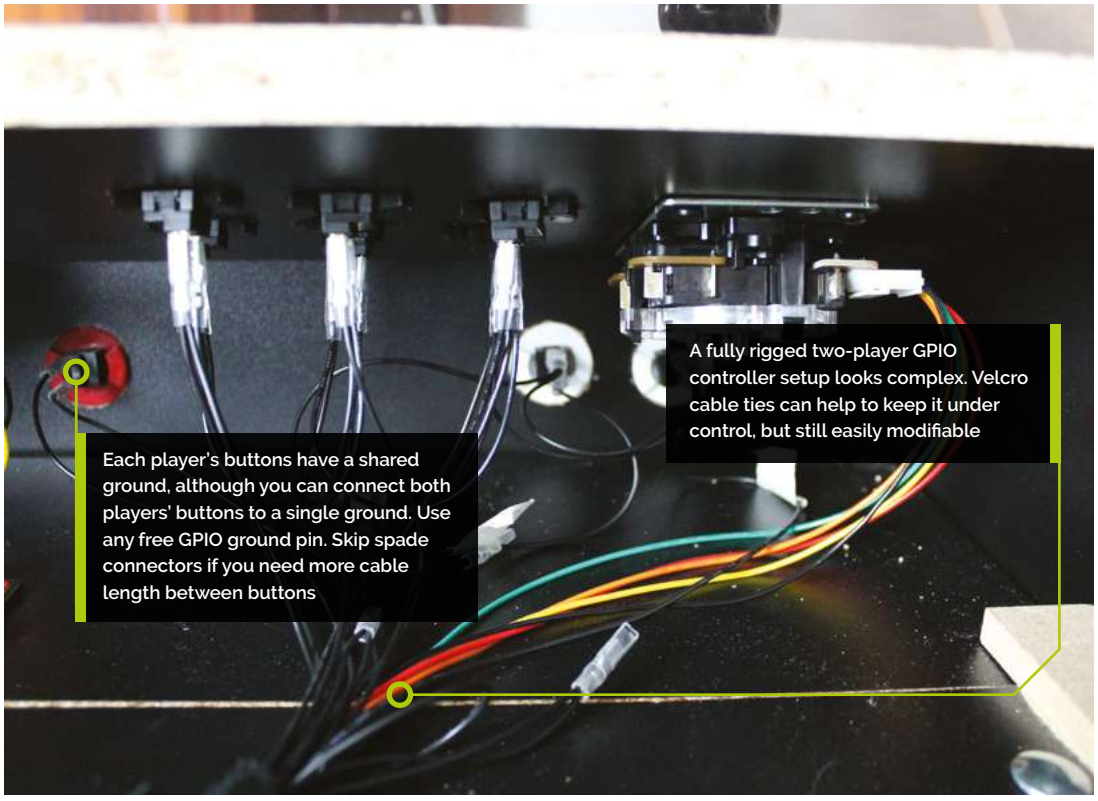
Recalbox's main menu takes a couple of button presses to get to but has a comprehensive set of configuration options

02 Connect to GPIO

This is the fiddly bit. We suggest using a case for Raspberry Pi that fully exposes the GPIO pins. The GPIO wiring diagram shows which buttons, directional controls, and ground connections should be attached to each pin. While buttons and controls can be reconfigured in software, ground cannot. Our setup uses a total of 25 GPIO inputs, plus four ground connections. Input 25 is for a dedicated hotkey button.

03 Install and power up

Open Raspberry Pi Imager, connect your microSD card writer, and Choose OS > Emulation and game OS > Recalbox and the version of Recalbox that matches your Raspberry



Pi. Click Write and wait for the image file to be written to the microSD card. When Imager has finished, remove the microSD card and insert it into the Raspberry Pi in your arcade build. Connect the cabinet's monitor and speakers to Raspberry Pi. Plug in a keyboard on a long cable. Plug in Raspberry Pi's power and it will boot to Recalbox's EmulationStation interface, which you can immediately navigate using the keyboard. However, we still have to enable our GPIO arcade controls, wireless networking, and other configuration options.

04 Connecting Recalbox

Recalbox has SSH and Samba enabled by default, as well as a web interface available via your browser on **recalbox.local**. Recalbox should appear on your network as RECALBOX (File Sharing).

> Recalbox has SSH and Samba enabled by default, as well as a web interface

A wired Ethernet connection will give you immediate access to these. If you don't have one, press **ENTER** to open the menu, scroll to Main Menu, and select it with **A** on the keyboard, then select Network Settings, enable WiFi, select your SSID, and then select 'Wifi Key' to enter your password. Recalbox only has a root user. The default username is **root** and the password is **recalboxroot**.

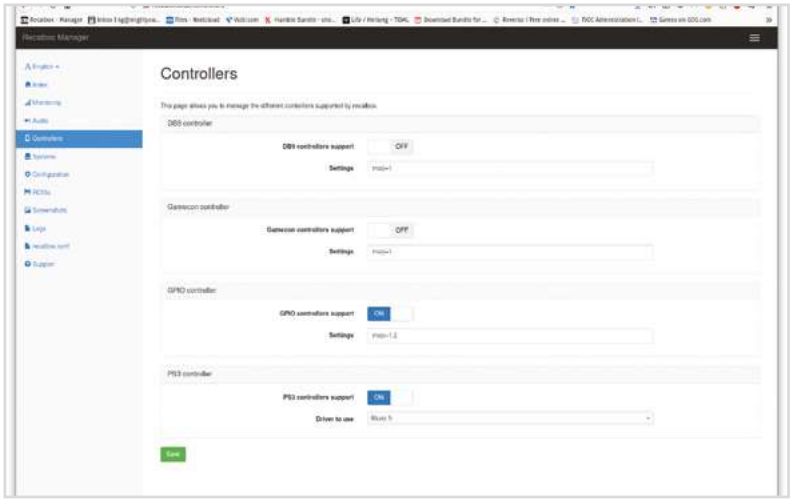
05 Configure Recalbox

You can access Recalbox's config file – **recalbox.conf** – by connecting via SSH, by browsing to the system directory in the Recalbox (File Sharing) Samba share, by pressing **F4** and then **ALT+F2** at the cabinet to exit to the console, or by going to **http://recalbox.local/** and selecting the **recalbox.conf** tab in the left-hand menu pane.

Under 'A – System options, Arcade metasystem', remove the semicolon that comments out `emulationstation.arcade=1`. This will make the arcade category the first entry in Recalbox's EmulationStation interface. →

You'll Need

- Spade to DuPont cables
- Spade to DuPont shared ground cables
- Joystick to DuPont cables
- At least one Neo Geo Classics Collection game **magpi.cc/ironclad**



A web interface at http://recalbox.local gives you control over almost every aspect of your arcade machine's setup

Under D2 – GPIO controllers, set `controllers.gpio.enabled=1`. Save your changes and, at the arcade cabinet, open the menu, go to Quit > Fast restart Recalbox.

Top Tip

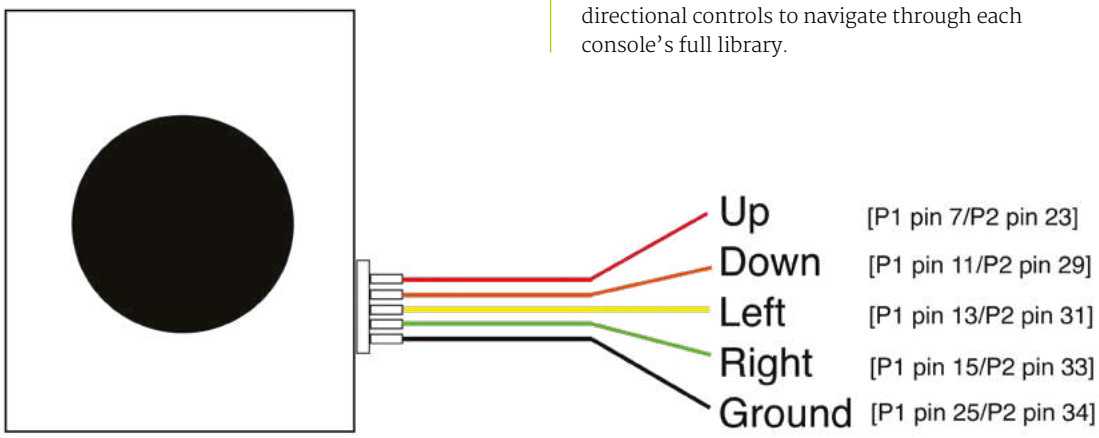
USB controls

To convert your controls to USB, use a Xinmotek board (**magpi.cc/xinmotec**) instead of connecting to GPIO.

06 Optional: Take control

Recalbox will now automatically detect GPIO controllers and, if all your buttons are wired as it expects, will already have the correct button configurations. Use the bottom-left button (B) to select options and the bottom-centre button (A) to go back. Left and right navigate between systems; up and down navigate between games or options within a menu. Press Start to open the configuration menu.

If your buttons aren't connected in that order, or if you prefer an alternative layout, open the menu and go to main menu > controllers settings > configure a controller. Press down to skip an entry that you don't have buttons for. If you don't have a hotkey button for one or more players, set it to Select.



Viewed from below, a standard Sanwa joystick's 5-pin connector goes to up, down, left, right, and ground. The diagram shows standard colour coding

07 Sounds good

If you have no sound, open the menu, select sound settings, and check the output device. We had to switch to 'headphones – analog' output to use our cabinet's speakers, connected to the 3.5 mm output on Raspberry Pi.

Recalbox plays background music all the time by default. This is charming, but a bit much for a cabinet that lives in the sitting room. Switch the Audio Mode to 'Video sound only' – to only hear the splash screens on boot – or 'No sound'.

If you prefer, you can add your own music by copying it to Recalbox's **share/music** directory.

08 Getting to know Recalbox

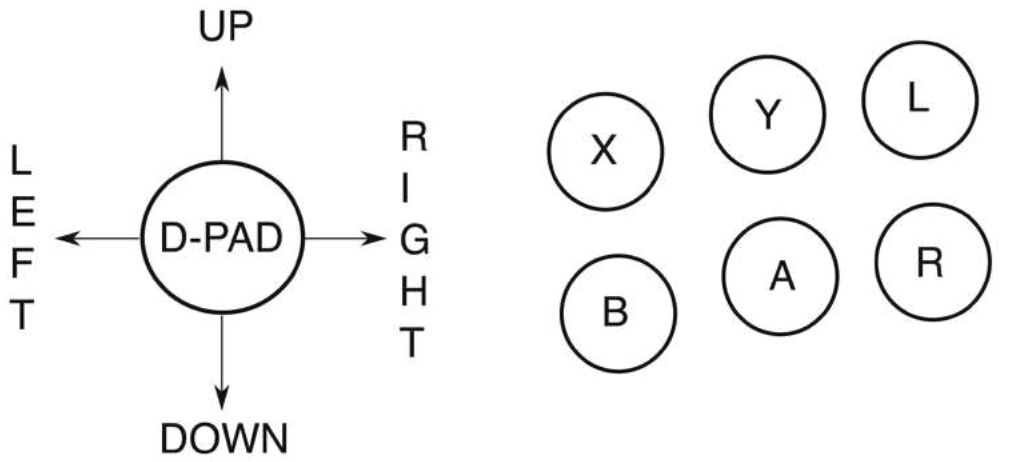
Recalbox comes preloaded with a number of freeware and open-source games. Because we enabled arcade mode, this category appears first. There are already four games loaded into it.

Select the category by pressing button B and scroll through them with the joystick. Gridlee, released in 1982, looks great for the era. Press B to load it.

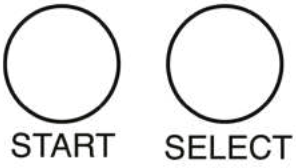
Press Select to add credits and press Start when you're ready to play. When you've had enough, press the hotkey button and Start together to quit back to the Arcade menu.

> "Press the hotkey button and Start together to quit back to the Arcade menu"

You can press A to go back to the top menu, and use the joystick to navigate up and down through the list. But it's easier to use the right and left directional controls to navigate through each console's full library.



Button and joystick correspondences for player controls. The joystick maps to the D-pad. L and R correspond to L1 and R1, equivalent to the shoulder buttons of modern joypads



Warning!
Mains electricity & power tools

Be careful when handling projects with mains electricity. Insulate your cables and disconnect power before touching them. Also, be careful when using power tools during this build.

magpi.cc/drillsafety
magpi.cc/electricalsafety

09 recalbox.local

Once your arcade machine is connected to your local network, you'll be able to access it in a web browser via **http://recalbox.local**. On the main page, you'll see shortcuts to a virtual gamepad, keyboard, and touchpad, which allow you to navigate through the arcade machine's menus remotely.

To add some authenticity to older titles, go to Systems and set the Shader set to Retro, which will apply community-favourite shader and scanlines settings to each game. On the other hand, if performance is poor, disable shaders and rewinding here. Click Save at the bottom of the page to store your changes.

Below, the Configuration tab lets you set networking options, enable and disable the Kodi media player, and configure the behaviour of the EmulationStation front end and the hotkey.

10 Manage game & BIOS files

The easiest way to manage your game ROMs on Recalbox is via the web interface, where the ROMs tab lets you select the directory for your desired console, stop the EmulationStation front end, upload games, and restart EmulationStation to load them.

You can also copy games over to the **roms** directory in Recalbox's Samba share. Even if you don't plan on emulating a specific console, don't delete the containing folders for its games, as they're required.

Recalbox also shares a **bios** directory, where you can add freeware or legally purchased computer or console BIOS files.

11 Buy and install a game

ROMs and a functional BIOS set for a number of Neo Geo maker SNK Corporation's games are available to buy as part of the Neo Geo Classics Collection (**magpi.cc/neogeoclassics**).

You'll need a Windows, macOS, or Linux PC to install or extract these. You'll find the ROM and BIOS files in the install directory; for example, **ironclad.zip** and **neogeo.zip** respectively for the fantastic scrolling shoot-'em-up Ironclad. If you don't want the whole collection, you can buy Ironclad alone at **magpi.cc/ironclad**.

Connect to Recalbox via SMB and copy the game ROMs into **roms**, and **neogeo.zip** into **bios**.

Restart EmulationStation and you should find your new games in the Arcade game list. Not all of them will work out of the box. Start any of them and press the hotkey and B buttons to open the Libretro emulation interface. Scroll down and select Options > Neo-Geo mode > Use UNIBIOS Bios. We aren't using UniBios here, but the file supplied by SNK is compatible with this setting. →

Top Tip

Preconfigured USB support

If your cabinet uses a USB controller board, then RetroPie won't need any extra drivers to detect your controls.

TUTORIAL

▲ SNK has made plenty of its arcade ROMs available to buy. Ironclad for Neo Geo-based arcade machines is a particular favourite

▶ GPIO wiring: Connect your joysticks and buttons to Raspberry Pi's GPIO as shown. *Image by digitalLumberjack of the Recalbox project, licensed under GPL2*

Press A twice to go back and select Resume. Your game should start.

12 Tweak your games entries

To hide the games that come with Recalbox, from EmulationStation press Start > Main menu > Games settings > Hide preinstalled games. Unfortunately, you can't pick and choose which get hidden, but you can manually download and re-add any that you'd like to keep.

You can also disable the ports category by editing **recalbox.conf** to include:

```
emulationstation.collection.ports=1
emulationstation.collection.ports.hide=1
```

If you want to add images or change the titles of the games you've added to Recalbox, the easiest approach is to use the built-in scraper. Highlight the game in the menu, press Start > Edit game > Scrape. You can also add your own ratings and keywords in this menu.

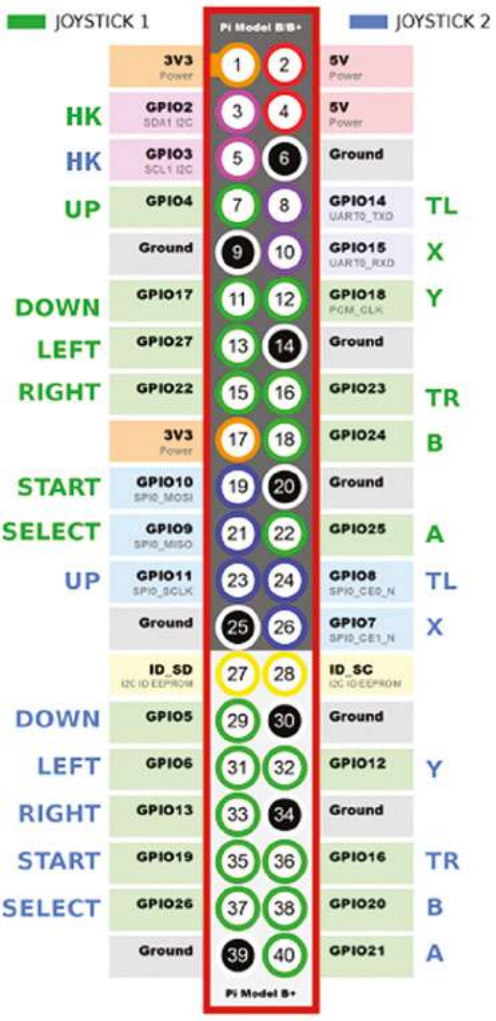
13 Get more games

The creators of the MAME emulator have been given permission to distribute some early arcade games, which you can find for download at **magpi.cc/mameroms**.

Many other emulated arcade games have been released for use on modern computers, but some – including collections by SNK, Capcom, Irem, and Namco – require an additional extraction and re-bundling stage. You can find tools and game lists to help you buy and use these at RED-project (**magpi.cc/redproject**) and SF30ac-extractor (**magpi.cc/sf30ac**). Linux GOG users may also require innoextract (**magpi.cc/innoextract**). Non-Neo Geo arcade games should go into the **roms/MAME** directory.

The homebrew scenes for arcade games tend to focus on physical releases, but we've had luck with Codename: Blut Engel for Neo Geo and Santaball (**magpi.cc/neogeohomebrew**) for Neo Geo CD.

For more retro and homebrew games that work well with arcade controls, including Sega's Mega Drive Classics collection, see **magpi.cc/legalgameemu** and **magpi.cc/legalroms**.



Warning!
Copyright alert!

It is illegal to download copyrighted game or BIOS ROMs in the UK without the permission of the copyright holder. Only use official purchased or freeware ROMs that are offered for download with the consent of the rights holder.

magpi.cc/legalroms

Wireframe

Join us as we lift the lid on video games

Visit **wfmag.cc** to learn more

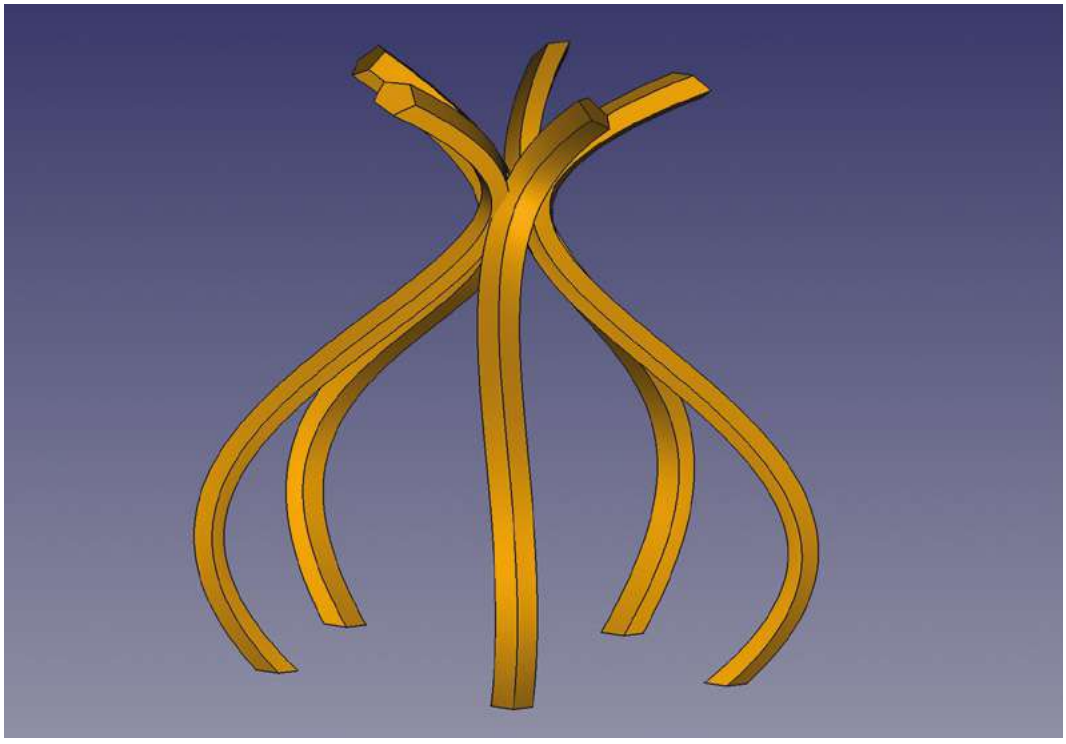
FreeCAD: making multiples of things

Save time with clones, arrays, and drafts

Jo Hinchliffe

@concreted0g

Jo Hinchliffe is a constant tinkerer and is passionate about all things DIY. He loves designing and scratch-building both model and high-power rockets, and releases the designs and components as open-source. He also has a shed full of lathes and milling machines and CNC kit!



YOU'LL NEED

A laptop or desktop computer

Figure 1 Creating arrays can produce fascinating geometries from a single copy

Often when we are designing something, we would like to make multiples of some aspect of the design and we want to do it as quickly and as efficiently as possible. A common example might be placing circles to become pockets, holes, or pads in a sketch. As a first example, let's go straight to the Sketcher Workbench and create a new sketch in any plane.

In **Figure 2**, you can see the desired sketch item on the left, a small square with a hole in each corner. To create this sketch, we drew four separate circles and constrained each of them dimensionally and positionally. It's a fine approach, but it's quicker to use some of the cloning and array tools the Sketcher Workbench offers. In Sketcher these tools can create matching objects and carry over some of, or manipulate, the existing constraints to save us some time. Starting over, if we draw one circle for example in the upper left-hand corner of the square, we can constrain the radius and then the position by setting a vertical and horizontal distance constraint between the centre point of the circle and the origin point of the plane. Next, select the circle and the Y axis datum line. We can now click the 'Create symmetric geometry with respect to the last selected line or point' tool icon. This should now create a second

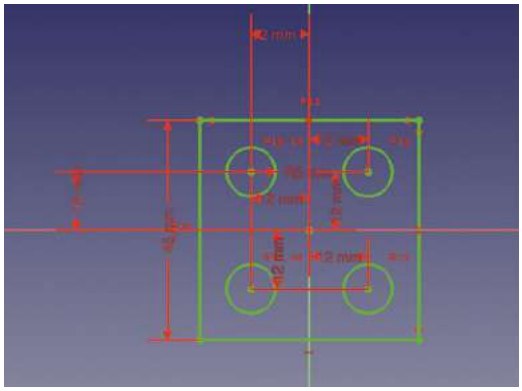
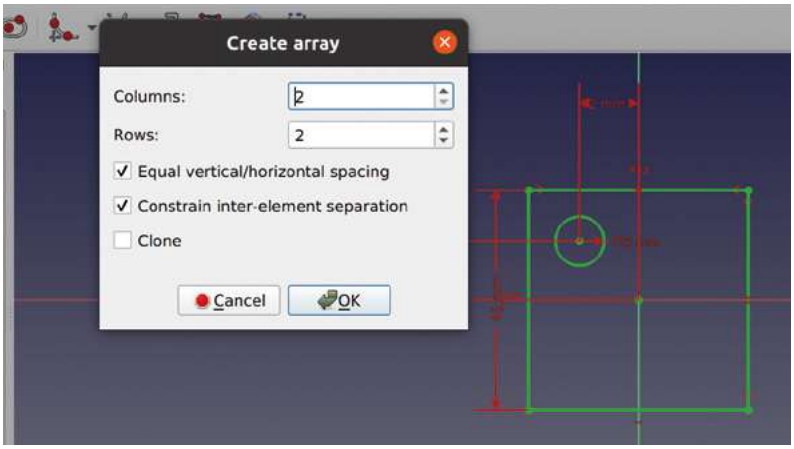


Figure 2
This symmetrical square sketch with circles can be created in numerous ways

circle, with the same radius constrained and in a mirrored position in the upper right-hand corner of our square. It doesn't have the positional constraints, but it is in the same mirrored position as the original, so if you add a vertical and horizontal constraint without changing the default position then they will remain correct. Of course, you can then select the two circles and the X axis datum line and repeat the process to create two more circles in the lower left and right corners.

Another approach is to use an array tool called 'Create a rectangular array pattern'. Starting again with just our upper left constrained circle highlighted, click this tool and you'll see a dialog box (**Figure 3**). In the dialog, set the column and row number each to '2' and then tick the 'Equal vertical/horizontal spacing' checkbox as well as the 'Constrain inter-element separation' box. Clicking OK, if you now move your mouse around the sketch area, you will see a line protruding from your original circle; this represents the angle at which your array will be created and the distance from the original circle. Left-click to place the rectangular array of circles into the sketch. Don't worry if you set it at a slight angle: click on the generated angle constraint and set it to zero degrees. Finally, edit one of the distance constraints it has added; as our original circle is 12 mm from the datum point vertically and horizontally, we set the constraint distance to 24 mm, which makes our square array of circles fully constrained and accurately placed.

For our next examples, let's move to the Part Design Workbench and create a body and then create a sketch in the XY plane. In the sketch, draw a simple rectangle and don't worry about constraining it. Close the sketch and pad the sketch to any thickness. Next, select the upper face of the padded rectangle in the preview window and click to create a new sketch on that surface. Select the 'Create a slot in the sketch' tool and draw a small slot at one end of



the rectangular surface. Close the sketch and then perform a pocket to cut the slot into or through the pad. Just to add to the example, select an edge of the slot and apply a fillet, which should auto-complete around the entire pocket edge.

Figure 3
Using the rectangular array tool to save time in sketches

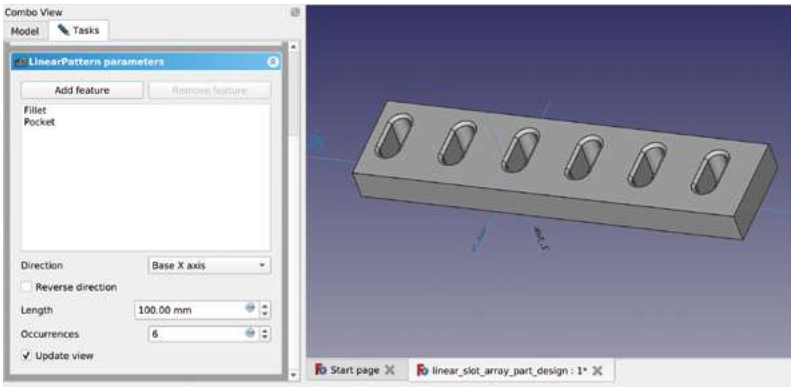
Figure 4
Creating a linear array of chamfered pockets

DESIGN PARTS WITH PART DESIGN

The Part Design Workbench has some specific array tools, and we are going to try the 'Create a linear pattern feature'. With nothing selected in the file tree, click the linear pattern tool. You should now have a dialog opened in the Tasks tab of the Combo View window. At the top of this dialog, you should see a panel titled 'Select feature' and all the available features should be listed; in this case, that is the pad, the pocket, and the fillet. For now, highlight just the pocket item in the list and click OK. You may see a duplicate pocket in the other end of your rectangle now, but don't worry if you don't. You should see that the dialog box in the Tasks tab has now changed; scroll down to the bottom of this new panel. You should see two adjustable parameters: 'Length' and 'Occurrences'. The value of Length represents the →

QUICK TIP

We covered the basics of sketching, padding, and pocketing items in the first two parts of this series, starting in issue 37.



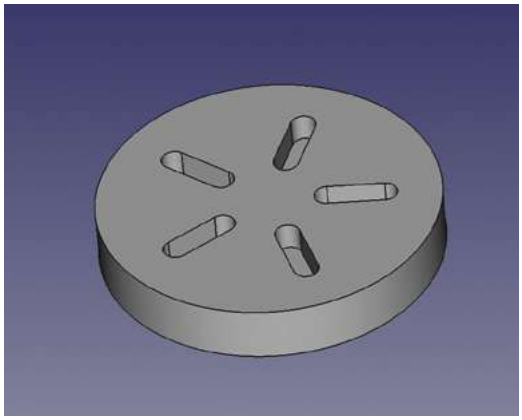


Figure 5 Creating a polar pattern of slots around a disc

length of the array and isn't linked to the length of any attached object. At the default length of 100mm with two occurrences, this might place the second occurrence of the pocket outside of the rectangular pad. Increase the number of occurrences to 5 and adjust the length of the array so that your array of pockets fits in the rectangle. You might also discover that if the length is shorter than the multiple widths of the pocket, it will overlap the pockets in the array. Before we close this example, you'll note that our fillet hasn't been replicated in the array. To add the fillet, click the 'Add feature' button, then either select part of the original fillet by clicking it in the preview window. Alternatively, you can swap to the Model tab and select the fillet in the file tree. Now, when you click OK on the linear pattern dialog in the Tasks tab, you will see the array with the fillets added to each instance of the pocket (**Figure 4**). Removing a feature is slightly counter-intuitive in that you can't select the feature to remove in the list in the 'Add/remove feature' box in the dialog. Similar to adding a feature, you click the 'Remove feature' button and then highlight the feature in the preview or in the file tree view.

Sometimes we might want to make an array of objects radially. As an example, start a new body and in a sketch on the XY plane, draw a circle with 30mm radius; constrain the centre of the circle to the datum point. Close the sketch and pad the circle. Similar to our earlier example, create a sketch on the top surface of our cylinder and draw a small slot to create a pocket. First, leave the slot without constraints and position it somewhere in the upper left quarter of our cylinder face and perform the pocket.

ROUND AND ROUND

Again, with nothing highlighted, click the 'Create a polar pattern feature' tool icon. The resulting dialog box is similar to our earlier linear pattern tool. Click 'Add feature' and select the pocket that we made. Scrolling down, leave the axis and angle settings as the default and set the occurrences to 5. If you have the 'Update view' box ticked, you should see that the pocket is replicated and rotated around the datum point. As our original slot was off axis, the array positioned the repeated slots relative to this original position; we can of course go all the way back to the root sketch of the pocket and adjustments will be recomputed to the pattern array. As shown in **Figure 5**, we closed the polar pattern dialog and opened the slot sketch in the file tree. In the slot sketch, we constrained the slot so that the centre of the slot was positioned on the Y axis datum line.

QUICK TIP

Press **SHIFT+V** or **SHIFT+H** to quickly add a horizontal or vertical distance constraint between two selected points or a selected edge.

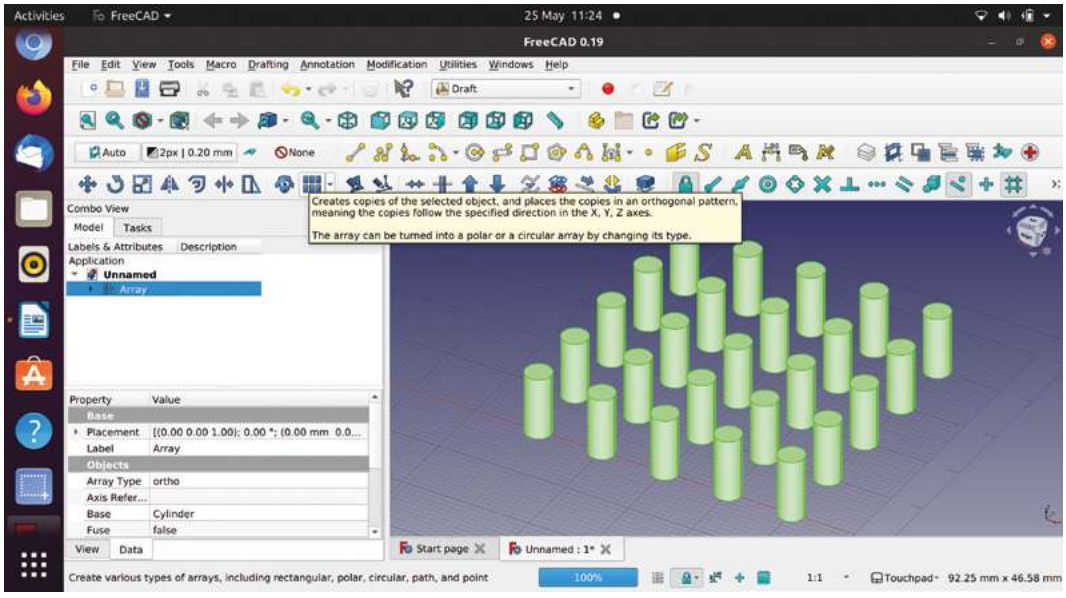


Figure 6 Simple array of parts using the rectangular array tool

Many times we will want to create more complex arrays, sometimes with more complex geometries or sometimes not in a single continuous body, which means we would be working outside of the Part Design Workbench. The built-in Draft Workbench has some array tools that can be extremely useful for this, but it's a useful Workbench for much more as well.

ON REPEAT

As a simple example, let's start a new empty project and move to the Part Workbench. Click the 'Create a cylinder' tool to create the default 10 mm tall cylinder with 2 mm radius. Next, use the drop-down menu to move over to the Draft Workbench. Let's highlight the cylinder in the file tree and then click the array tool icon, which resembles a collection of six blue rectangles. In the dialog box, the first items you can adjust are the 'Number of elements'; edit the X and Y values to 5 and leave the Z at 1. This is then aiming to create a rectangular array of 5×5 of the cylinder object. Next, we can edit the spacings of the array. It's slightly counter-intuitive in that the X, Y, and Z elements each have an X, Y, and Z input box, but if

> Many times we will want to create more complex **arrays, sometimes with more complex geometries**

you play with the values and apply the array, you'll soon see what effect these have. For now, in the 'X intervals' section, set X to 10 mm. In the 'Y Intervals' section, set Y to 10 mm and leave the Z, as we only have one layer in our array. Clicking OK, you should now see an array of our cylinders spaced at the intervals we set (**Figure 6**).

Highlight the array item in the file tree and delete it. We should now see our cylinder item and if we highlight it and press the **SPACE** bar, we'll make it visible again. Next, let's try the 'Polar array' tool from the Array tools drop-down menu. In the dialog, leave the polar angle set to 360 degrees to create a full circle array and set the number of elements to 5. We need to add a point which is the centre of rotation; if we left the centre of rotation set at the 0,0,0 point, we would simply create five cylinders placed on top of each other. Change the Y co-ordinate to 10 mm and then click the OK button. You should now see a circular array of five cylinders appear.

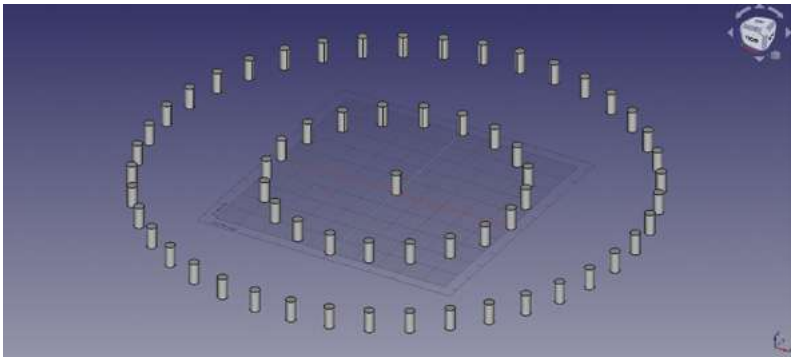


Figure 7
Creating circular arrays of Part objects

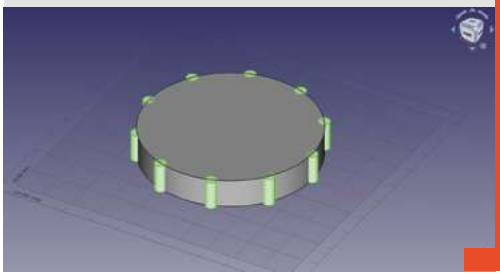
Delete the polar array and let's try the circular array tool. Highlight our test cylinder and click the 'Circular array' tool icon. This tool works differently in that the number of copied objects in the array is dependent on the size and spacing of the array rather than a directly input value.

In the dialog box, the 'Radial distance' is the radius of each circle in the created array; we set this to 50 mm for a test. If we don't change the centre of rotation and leave it at the datum or zero point, then the first circle in our array will be made with a →

FOLLOW THE PATH

Sometimes we will want to create an array of objects, but constrain them onto another object or path. This can be achieved using the path array tool. In a new project, once again go to the Part Workbench and create a first cylinder and set the radius to 30 mm and leave it at 10 mm tall. Then create a second cylinder and leave it with the default dimensions of 10 mm tall and 2 mm radius.

Move back to the Draft Workbench and first select the object of which we wish to create an array; let's select our second, smaller cylinder. Next, we select the path we want the array of objects to be connected to; holding **CTRL/CMD**, click the lower outer edge of the larger cylinder. Then click the 'Path array' tool; this applies an array without a pop-up dialog box, but the resulting array has some editable parameters in the Combo View window. You should see the array has been created with four instances of the smaller cylinder placed equidistantly around the larger cylinder edge. If we highlight the array object in the file tree and adjust the 'Count' parameter, it will add or remove instances from the array and redistribute the count to maintain equal distances between each instance.



QUICK TIP

Near the Part Design pattern array tools, you might notice the 'Create a mirrored feature' tool, which is similar to the symmetric geometry tools on the Sketcher Workbench.

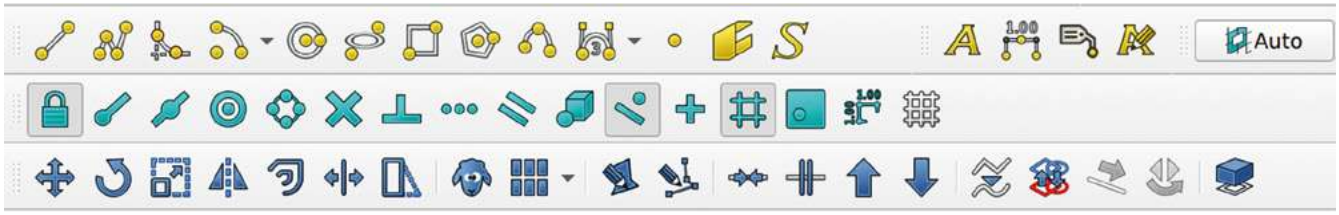


Figure 8
Some of the tool icons on the Draft Workbench: top row drawing tools, middle row snap options, and lower row includes array tools

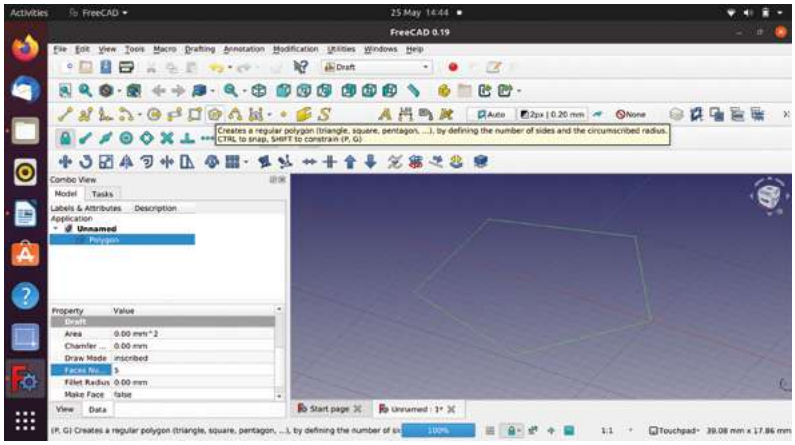
50 mm radius from this point. The second circular layer will be made at a further 50 mm from the first circle so therefore will have a 100 mm radius, etc. Next, the 'Tangential difference' is the spaced distance between instances of the array object in the same circular layer of the array. We initially set this to 15 mm, but play with this value to see how it affects your test arrays. The 'Number of circular' is the number of concentric circles that are added, spaced as we said at the radial distance; we set this to 3 for our test. Finally, there is the Symmetry input; it's difficult to explain the effect, but this value controls the number of symmetric planes an array has – the best thing is to start with it set to 1 and then change the values and see what effect it has on your array! With the above settings, you should see an array similar to **Figure 7**.

We've seen that the Draft Workbench array tools can be used to create arrays of copies of a part, but they can also be applied directly to sketches and even be applied to bodies created on the Part Design Workbench. The caveat with arrays of a Part Design body is that the array will usually create separate objects that cannot be considered part of the original body. When this occurs, the array is listed outside of the active body in the file tree and the original body part visibility is toggled off.

As a final example exploring arrays, let's create a complex-looking decorative array using other Draft Workbench tools to create the underlying geometries.

Figure 9
Drawing a pentagonal polygon in Draft is similar to the Sketcher Workbench apart from not using constraints

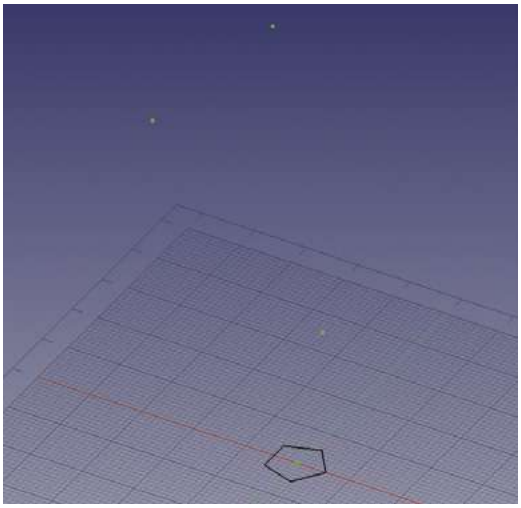
Figure 10
Creating points to act as snapping points for a B-spline



Create a new project and move to the Draft Workbench. You should see a grid on the XY plane. If you don't, find the 'Toggle Draft Grid' icon, which looks like a mesh and is at the end of the snap toolbar icons; toggle this button until you see a draft grid.

You may notice that the Draft Workbench has a collection of drawing tools that look similar to the Sketcher drawing tools. They do work in very similar ways, but in the Draft Workbench, wires and edges are drawn without constraints and use either snapping to the grid or other objects, or inputting positional co-ordinates. As such, the Draft Workbench has a lot of options for different snapping styles. For our example, we need to enable snapping by clicking the 'Snap Lock On/Off' padlock icon. This should now activate the other snap icons; we want 'Snap grid' and 'Snap near' to both be on, as seen in **Figure 8**.

First, let's draw a polygon on the Draft grid so that it sits on the XY plane. Click the 'Create a regular polygon' tool icon and then left-click to draw a polygon around the datum or zero point. To help you draw it accurately, zoom in towards the datum point and you should see the cursor lock to the grid before you click to draw. Adjust the polygon placing circle to roughly 6 mm in diameter. In the properties dialog, you can adjust the number of sides, but as a default it is set at three. If you create the default triangle, you can



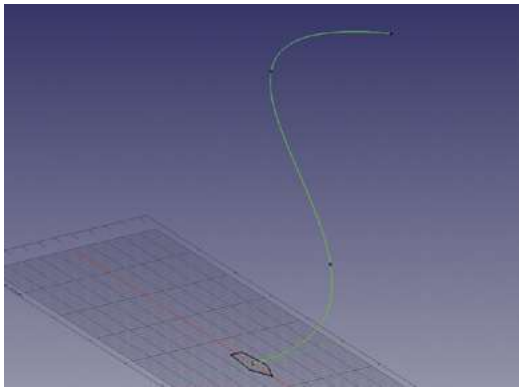


Figure 11
Using the B-spline tool to draw a curved line between the points we created

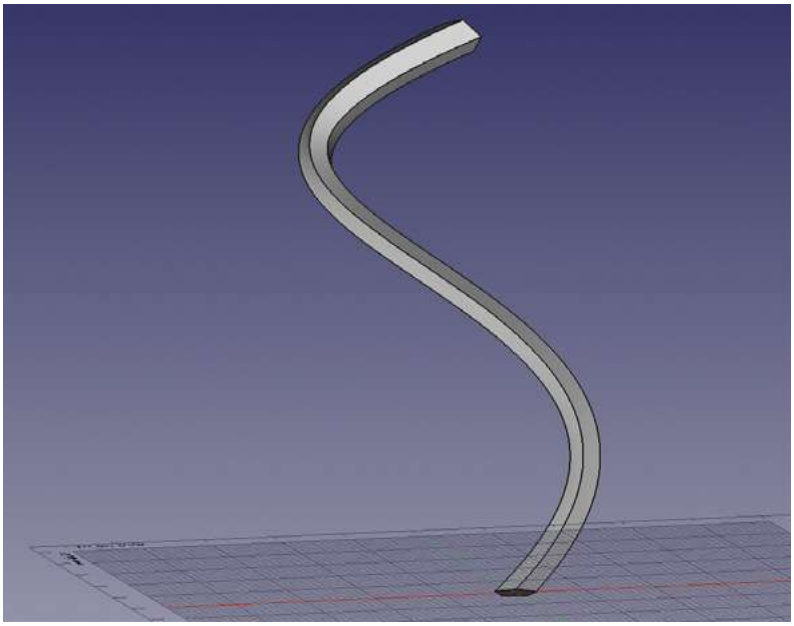
highlight the Polygon object in the file tree and adjust the number of sides in its properties table.

CURVED LINE

Next, let's create some points which we will use to create a curved line in three axes. The first point of the line is going to be the centre of our polygon, or the zero point. Click the 'Create a point object' and click to place a point at the datum. Next, let's add a second point and then raise it on the Z axis. We are aiming to create a curve, so place this second point on the XY grid at a random point close to the zero but not on it. We went with an X co-ordinate of 6 and a Y co-ordinate of -2. Having created this second point, we highlighted it in the file tree and, in the dialog, adjusted the Z axis position to 30mm. You may need to adjust the zoom, but you should see this point now rise up above the plane. We added two more points above: the next had the co-ordinates X = -28, Y = 0, Z = 60, and the final point was at X = -10, Y= 9, Z = 80 (**Figure 10**).

Creating a curve between the points we just made is easy with the 'Create a multiple-point B-spline' tool. Select this tool and hover near the upmost point we just made. When you are hovering, if you are too far away from the raised point, you'll see that next to the pointer position there is an icon that looks like the Draft grid icon; this indicates that if you click, you'd be snapping to the grid. If you are close enough to the raised point, this icon disappears, and if you left-click, you will attach the B-spline to the point. Left-click on the upper point and then move to the next lower point and left-click again; continue to do this until you reach the last point at the zero position. Left-click to attach the last point and then click the 'Close' button (**Figure 11**).

Just for a moment, we are now going to switch to the Part Workbench to use the sweep utility. On the Part Workbench, click the 'Utility to sweep' tool icon and in the dialog, select the Polygon item we drew from the left-hand list, and click the right-facing arrow to add it to the right-hand column. Next, click the 'Sweep Path' button, then left-click on the B-spline curve we drew in the preview window and click 'Done'. Check the 'Create solid' box and click OK. You should now have a curvy swept hexagonal object in the preview (**Figure 12**).

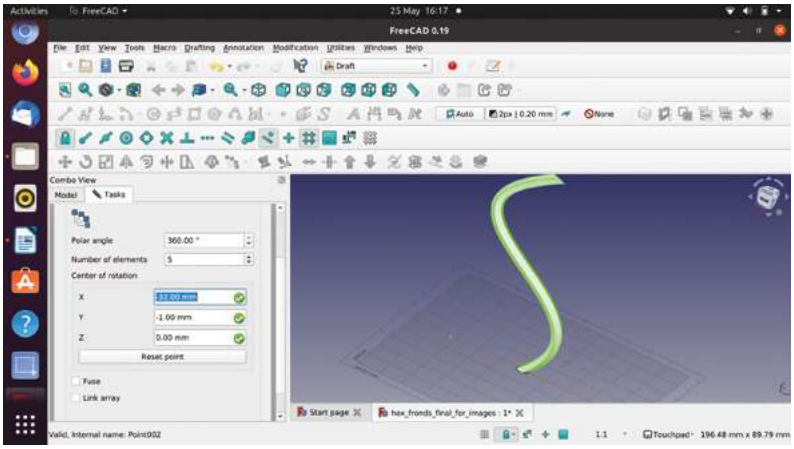


Moving back to the Draft Workbench, select the swept object in the file tree and then click the polar array tool icon. You can then left-click in the preview window to select a centre of rotation (**Figure 13**). We put our centre of rotation at around -30mm on the X axis to create our array of five of our swept items; you can see the completed array in **Figure 1**.

Hopefully you've found this article interesting and it's shown that the variety of cloning and array tools are useful in accurate technical modelling, but also in more playful experimental modelling. We look forward to seeing what you produce using these techniques. ◻

Figure 12
Using the sweep utility on the Part Workbench to create the final part for our array

Figure 13
Selecting a centre of rotation for the array of swept items; we chose -30 in the X axis



TUTORIAL

Upgrade your laser cutter with an adjustable bed

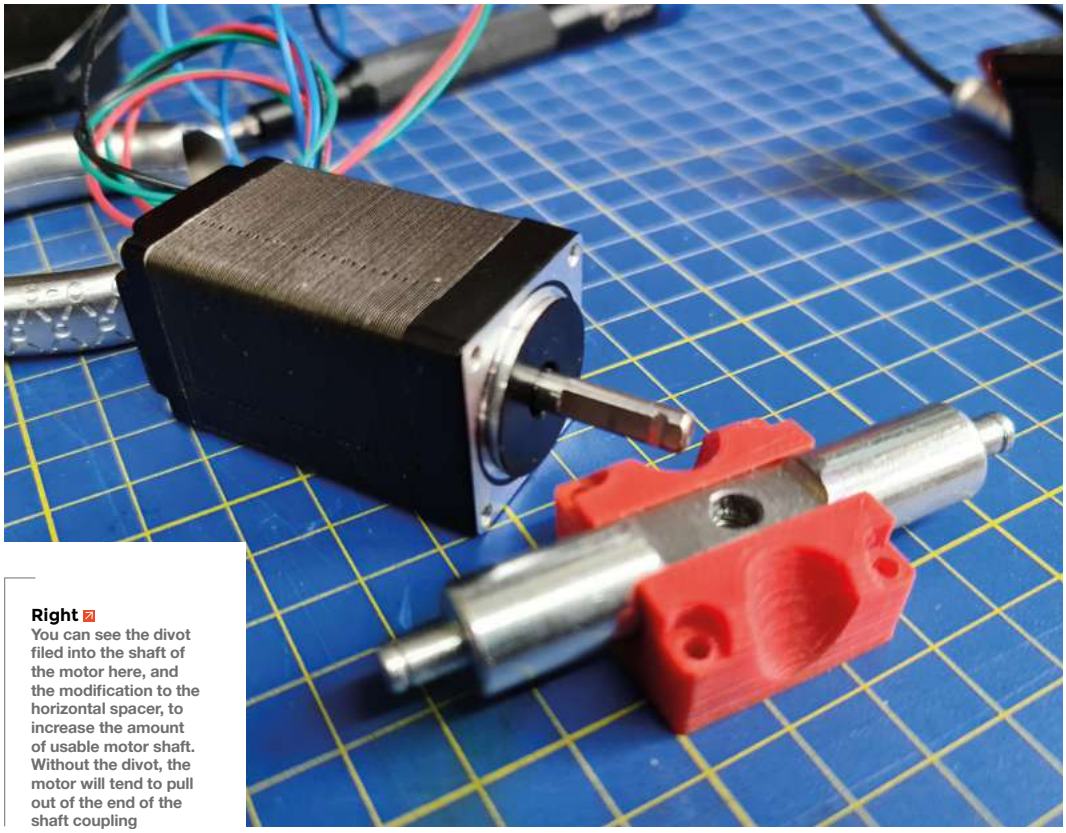
Enhance your laser cutter using an Arduino Uno, a stepper motor, and some push-buttons

Dr Andrew Lewis

Dr Andrew Lewis is a specialist fabricator and maker, and is the owner of the Andrew Lewis Workshop.

One of the biggest drawbacks of owning a K40-style laser is that the beam has a fixed focus and a fixed bed. Without the ability to focus the laser, you'll never be able to get the best out of your K40 when cutting with thicker materials. While you could replace the laser head, you'll probably find that the replacement costs considerably more than the whole laser cutter did. In this project, you'll learn how to make the next best solution: an adjustable bed. The idea behind installing a moving bed is that, while you can't move the focal point to the object, you can move the object to the focal point. Moving the machine bed is a much lower-cost solution that uses mostly off-the-shelf parts.

The motorised bed has three main parts: a control box, a motorised scissor lift, and the machine bed. The machine bed is made from 3 mm aluminium and has spikes arranged across the surface to minimise bed contact. The scissor lift mechanism is made from a modified laboratory jack. The lab jacks use a 6 mm threaded bar with a reverse thread on one side. Begin modifying the jack by disassembling the scissor mechanism and removing this threaded bar. The threaded bar passes through a horizontal support on each side of the jack. One of these supports will have a left-handed thread. Take the left-handed bar and file it flat on one side, perpendicular to the hole running through it. You will be fitting the motor's shaft through this hole, and adding a flat section on the inside increases the amount of shaft



Right
You can see the divot filed into the shaft of the motor here, and the modification to the horizontal spacer, to increase the amount of usable motor shaft. Without the divot, the motor will tend to pull out of the end of the shaft coupling

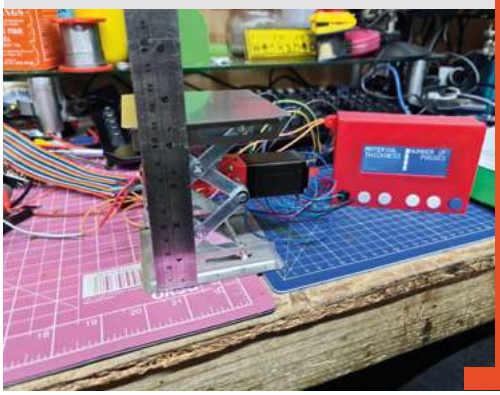
available for the threaded bar to grip onto. You'll need to file a depression into the motor shaft near the end, so that a grub-screw will grip the shaft without sliding off the end. Once you have done this, add the 3D-printed motor mount to the left-handed horizontal support, and then add the 3D-printed button mount to the other horizontal support. Reassemble the scissor jack, and mount the motor to the 3D-printed mount using M2.5 10 mm screws. Mount a push-button into the 3D-printed button mount. The button works as an end-stop, to prevent the jack from over-tightening. Tap the grub-screw holes in the side of the 3D-printed shaft coupling to accept M3 thread, and then screw the 90 mm M6 threaded bar into the end of the coupling. Screw the bar in place into the right-handed horizontal support, then push the motor shaft into the other end of the coupling, and hold it in place with a grub-screw.

Assemble the control panel by adding the buttons to the 3D-printed button bar, and then gluing them in place behind the front panel. Screw the LCD into place. Wire up the Arduino using the provided schematic, and flash it with the bed controller application. You can download this from **hsmag.cc/issue44**.

The new laser cutter bed is made from aluminium and has a lot of metal spikes screwed through it. Mark out a grid on the aluminium where you →

CONTROL **APP ARRAYS**

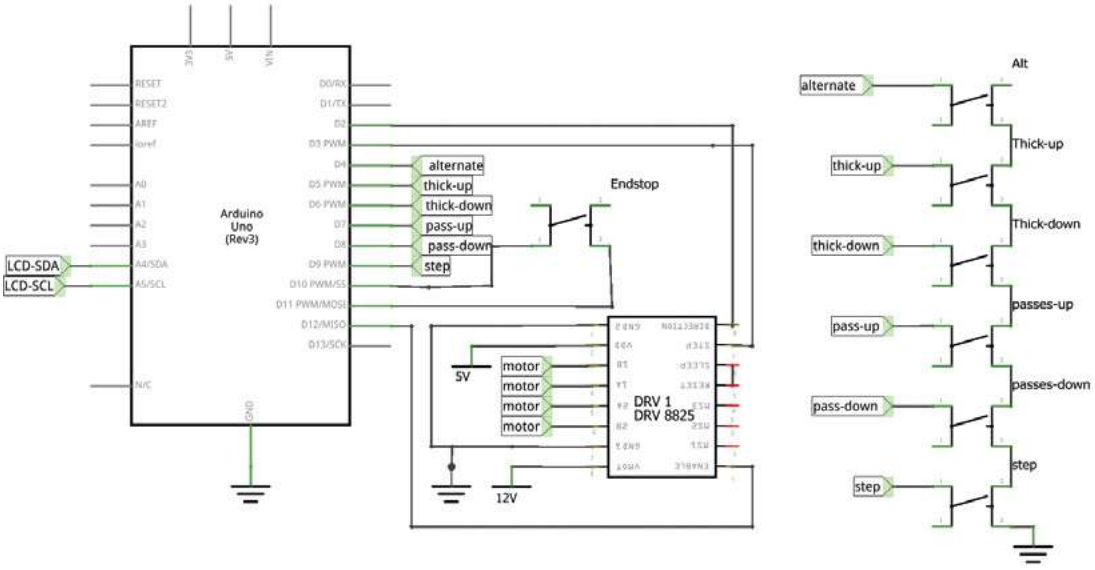
The top lines of the control application define two arrays called `mapIn` and `mapOut`. These arrays are used by the `MultiMap` function to help calculate the height of the scissor jack for a given number of stepper motor steps. The `mapIn` array is a list of stepper motor positions, and the `mapOut` array is the height that corresponds to that number of steps, measured in the real world. You can increase the accuracy of your turntable by tweaking these numbers to match your own setup. Create a function to move the bed to the end-stop, then raise the bed by increments of 100 steps. Jot down the measurements and tweak the `mapOut` table accordingly.



YOU'LL NEED

- **A4-sized 3 mm aluminium sheet**
- **1 length 10 mm square aluminium angle**
- **11 m 4 × 6 mm threaded insert**
- **2 m 5 × 10 threaded insert**
- **100 × 14 mm metal cones**
- **10 mm scissor jack**
- **NEMA 11 stepper motor**
- **90 mm of 6 mm threaded rod** or equivalent length bolt
- **Arduino Uno**
- **12 V power supply**
- **DRV8825 stepper motor controller**
- **20 × 4 LCD screen with I2C interface**
- **7 × 12 mm panel mount NO push-buttons**
- **4 × M2.5 10 mm screws**
- **3D-printed parts**

Left
While it's possible to calculate the theoretical relation between the number of steps and the scissor jack height using high school maths, it's actually more effective to take real-world measurements and interpolate from a table of known values



Right
The schematic for the controller is quite simple, using easily available parts. If you can find room, it might be possible to integrate the display and buttons into the existing controller box on the K40, but making the system easy to remove increases the overall flexibility of the laser cutter

Below
You can see the end-stop here, mounted on the horizontal support of the scissor jack. If you're worried about the duty cycle of the switch in the end-stop, you could replace it with an optical sensor

want your spikes to be. The layout is partly down to personal preference, although a 30 mm square grid is a good starting place. You may want to increase the number of spikes on part of the bed, to add extra support for small parts. The corner where the laser will hit its home position is a prime place to add some extra support. Drill and de-burr all of the holes, and then countersink them on the back. This is a boring, repetitive job, but worth the effort in the long run. Screw the spikes in place and then assemble the 3D-printed adjustable guide for the corner of the bed. The clamp is designed to provide a rigid corner alignment and multiple fixing points with threaded inserts, so you can design and make your own jigs to hold special parts. The two adjustable arms on the clamp have a recess on the corner to accept 10 mm aluminium angle pieces. These pieces will protect the printed clamp from stray laser light.

Before you fit the new bed to the laser cutter, you'll need to remove the existing bed. The exact fixing method varies between different flavours of laser cutter, but most will be screwed to the rail or base of the machine. Save yourself some time by marking the current height of the bed on the edge of the machine somewhere, before you remove

CALIBRATE!

The DRV8825 motor controller makes controlling a stepper motor very simple, but it isn't quite plug and play. You need to calibrate the maximum output of the controller, otherwise, it could overheat or burn out the motor. To calibrate the DRV8825, you need to use a multimeter to measure the voltage between the potentiometer and ground pin. The voltage needs to be set (at most) to half of the maximum current of the motor you are connecting to it. So, for a 0.6 A motor, the voltage should be 0.3 V. It's wise to aim a little bit lower than the theoretical maximum, to be extra safe.

the bed. That way, you'll have a guideline for where you need the height of the new bed to be when you're calibrating.

Fix the spiked bed to the top of your scissor lift. A single scissor lift fixed at the middle of the bed should be adequate for most needs, but if you are intending to work with heavy loads on the bed, you might find that building a second jack and fitting one at each end of the bed will give you more lifting power. The intended lifting duty of the bed will also affect the method you choose to fix the bed to the jack, and the jack to the base of the laser. Drilling and bolting through the bottom of the case and the base of the jack will give you the most rigid solution, although contact glue, carpet tape, or even magnets could give you a quick fixing solution. Gluing both ends of the jack is not recommended at all, as it will be very difficult to remove or adjust the bed without damaging the laser.

Routing for the motor and end-stop cables will depend on where you intend to put the control box. If you're mounting the box at the front of the machine,

> Save yourself some time by marking the current height **of the bed on the edge of the machine somewhere**

then you'll need to either make a cable hole at the front, or run the cables under the machine and in through a hole at the bottom. The table needs a 12V power supply to run. If you have already fitted a secondary power supply into your laser cutter, then you could power the controller from that line using the VIN pin on the Arduino and the VMOT connector on the DRV8825. If you prefer a separate wall wart-style supply, then you can make a hole in the bottom of the control panel and plug the supply directly into the Arduino power socket, with internal wiring running from the Arduino power socket to the DRV8825.

With the new table fitted and calibrated, you should notice an improvement in your cut size and find that the back of your material should be cleaner when it's cut. Unfortunately, the K40 is quite cramped inside and there isn't room to fit a measuring device with this design. If you're scaling the design up and have some extra room, you might find it easier to add a digital calliper to the bed and read values from that with the Arduino to get accurate height readings.

BUTTONS EXPLAINED

The heart of the control box is an Arduino, connected to a stepper motor controller and an LCD. The LCD displays the current height of the bed and the number of passes that you want to make with the laser. The left-most button works like an alternate key, and holding this button down will affect the other keys. Keys two and three increase and decrease material thickness in millimetres. Holding down the alternate key lets you move in tenths of a millimetre. The next two buttons increase or decrease the number of passes you want to make with the laser. Using the alternate button with these two buttons lets you move the bed up and down directly, ignoring the material thickness settings. It also means that you can set the focus point of the laser. Set the height of the bed to the ideal focal point, then hold down the alternate button and the sixth button. This will store the height of the bed in EEPROM so that it will return to this position (minus the material thickness) whenever the machine is restarted. Pushing the last button without the alternate key will manually step the bed to the next height in the sequence. So, if you set your material thickness to 3.0mm with two passes, pushing the button will reposition the bed to 1.5mm and update the material thickness.

QUICK TIP

You can control the pass button automatically, by altering the Python code for K40 Whisperer to trigger a GPIO pin when moving between passes.

Above
Adding spikes to your print bed reduces the chance that smoke, fumes, or other impurities will leave marks on the back of the object you're cutting. The location of the black areas on the edge of this project correspond to the location of the mesh used to support the wood

Left
The bed controller does more than just move the bed: it will automatically calculate heights for you on cuts that require multiple passes, and it can be integrated with K40 Whisperer software on a Raspberry Pi

Talk to your devices from your web browser

Discover how to use JavaScript and your web browser to interact with embedded devices

Rob Miles

@robmiles

Rob Miles has been playing with hardware and software since almost before there was hardware and software. You can find out more about his so-called life at **robmiles.com**.

In this article, we are going to discover something magical about modern desktop browsers: they can host JavaScript programs that can talk to serial ports in external devices. We're going to explore how to use this and then go on to create a browser-connected environmental sensor.

THE SIMPLETERM PROGRAM

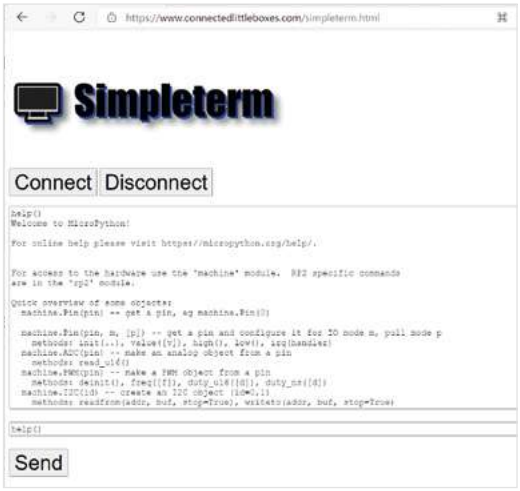
The best way to investigate how JavaScript applications can use the serial ports on your computer is to look at one created by the author. You can find the Simpleterm program at **hsmag.cc/Simpleterm**. It implements a very simple serial terminal, which takes in your commands and sends them to a device plugged into the serial port on your computer.

STARTING THE TERMINAL

When the user clicks the Connect button in Simpleterm, the first thing the program must do is determine which port to use and then create a connection to the device on that port.

```
this.port = await navigator.serial.requestPort();
await this.port.open({ baudRate: 115200 });
```

The first statement requests a port from the browser. At this point, the browser asks the user to select a connection using the dialog box shown in **Figure 1**. The second statement opens the connection with the specified baud rate. The baud rate specifies the rate at which characters are sent down the serial connection. It is important that the sending and receiving devices use the same baud rate. The rate of 115200 is commonly used with embedded devices. It is used by MicroPython and also by Connected Little Boxes.

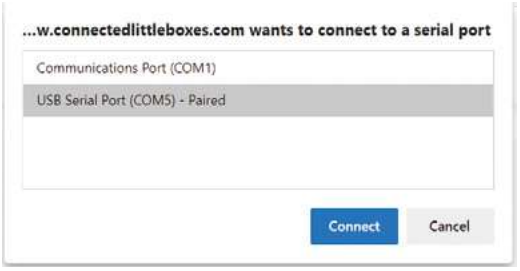


Right
Simpleterm is talking to a Raspberry Pi Pico device running MicroPython, which is plugged into my computer. We can enter commands into the lower text area and send them to the device by pressing the Send button. Messages from the device are shown in the upper text area

DATA DELIVERY

```
this.reader = this.port.readable.getReader();
// get a reader
const { value, done } = await this.reader.read();
// read data
if (done) { // check if we are done
    break;  // break from the read loop if finished
}
this.handleIncomingBytes(value); // deal with the
data
```

The code above shows how Simpleterm reads data from the serial port. The `port` object provides an attribute called `readable` that exposes a `getReader`



method that is used to get a **reader** object. The **reader** object provides a **read** method that is used to read from the port. The result of a read is a flag called **done** which is set to **true** if the read has been ended and a block of data called **value** which contains the bytes received from the port. The **value** object is passed to a method called **handleIncomingBytes** which deals with bytes received from the serial port.

```
handleIncomingBytes(bytes) {
    var text = new TextDecoder("utf-8").
decode(bytes);
    this.handleIncomingText (text);
    return;
}
```

A serial port sends data as a stream of 8-bit bytes. The **handleIncomingBytes** method uses a **TextDecoder** object to convert these bytes into a string of text. It then sends this text to another **rmethod** (called **textDestination**) for display on the web page.

```
function handleIncomingText(text) {
    let output = document.getElementById('output');
    output.value = output.value + text;
    output.scrollTop = output.scrollHeight;
}
```

At last, we are going to display the received text on the screen. The **handleIncomingText** method finds the output document element in the HTML document, adds the received text on the end of the text in the element, and then scrolls the element down so that newly arrived data is shown at the bottom of the screen. The process of fetching data from the port is performed in a loop so that bytes are repeatedly fetched from the serial connection and displayed in the browser.

SENDING DATA

```
async function doSend() {
    let input = document.getElementById('input');
    let text = input.value;
    text = text + '\n\r';
    await terminal.sendText(text);
    input.value = "";
}
```

SERIAL SECURITY

Before code running in a web page can access a serial device on a host computer, the browser will ask the user to confirm the connection. In addition, the serial communication features will only work at local sites (ones held in files on your computer) or websites hosted on sites that use HTTPS (the secure web protocol) for their connection. I've found the best way to get a secure website hosted for free is to use GitHub web hosting.

The **doSend** function is bound to the Send button and runs when the button is pressed. The function finds the input element in the web page, gets the text out of it, adds a linefeed and carriage return to the end of the text, and then calls the **sendText** function to send the text out of the serial port. Before a string of text can be sent out of the serial port, it must be converted into a sequence of 8-bit values.

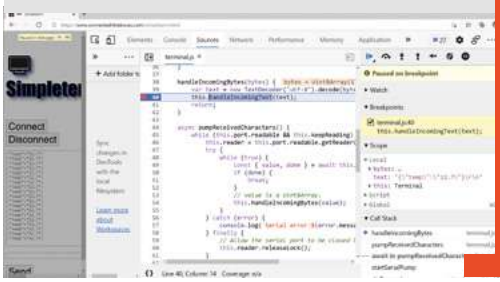
```
async sendText(text) {
    let bytes = new TextEncoder("utf-8").
encode(text);
    await this.writeUint8Array(bytes);
}
```

The **sendText** method above uses a **TextEncoder** to encode the text into bytes which are then sent by the **writeUint8Array** method.

```
async writeUint8Array(valArray) {
    const writer = this.port.writable.getWriter();
```

VIEW CODE IN THE BROWSER

Modern browsers contain powerful developer tools. You can open the Developer View of any web page by pressing **F12** on your keyboard. If you click the Sources tab in the Developer View, you can then open the files that power the website and look at them. You can even set breakpoints and step through the code to discover exactly how it works.



YOU CAN USE THIS WITH

- **A device that uses a USB serial connection**
- **A browser based on the latest Chromium platform, including Google Chrome and Microsoft Edge running on a desktop or tablet computer**
- **If you want to make the connected sensor, you'll need an ESP8266 or an ESP32, and a BME280 sensor**

Figure 1
Selecting a serial port from your browser. You may have to experiment to find the port where your device is connected

Left
Simpleterm dealing with an incoming message from the environmental sensor. It's been converted into a string of text to be sent to the display


```
    await writer.write(valArray);
    writer.releaseLock();
}
```

This method gets a write from the port, uses it to write the array passed as a parameter, and then releases the writer. This is the point at which the message is sent over the serial port to the destination device.

WRITE SERIAL CODE IN THE BROWSER

If you want to create your own web pages that use the serial port from the browser, you can find all the code inside the files that make up the Simpleterm program on GitHub: **hsmag.cc/CLBproject**. The two files that are most interesting are the **simpleterm.html** and **terminal.js** files. You can also use the debugger inside the browser to discover how the program works. You can host pages containing JavaScript programs on your own computer or laptop. I strongly recommend the Live Server plug-in for Visual Studio, which will host websites for you on your machine.

CREATING AN ENVIRONMENTAL SENSOR

Now that we know how to connect a browser to a device, let's make a device for the browser to talk to. We are going to create a simple temperature display page that runs inside the browser using a Connected Little Box as the sensor. The page is at **hsmag.cc/TempViewer**. When this page is connected to a serial device, it receives each temperature, decodes the JSON to extract the temperature information, and then displays it.

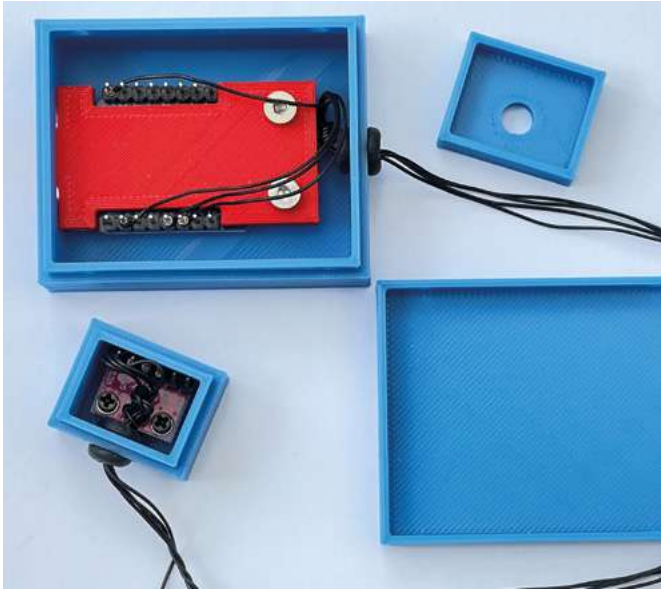
We've used a BME280 sensor, but it should be easy to adapt to a different sensor. You can connect it on a breadboard, but this author likes using wire-wrapping.

The sensor runs the Connected Little Boxes software. You can find out more about this in issue 41 of HackSpace magazine. We can configure the box to send a JSON-formatted temperature value every second by using the following command:

```
{"process":"console","command":"reportjson",
"text":"starting","sensor":"bme280","trigger":
"tempsec", "attr":"temp","store":"start","id":
"temp"}
```

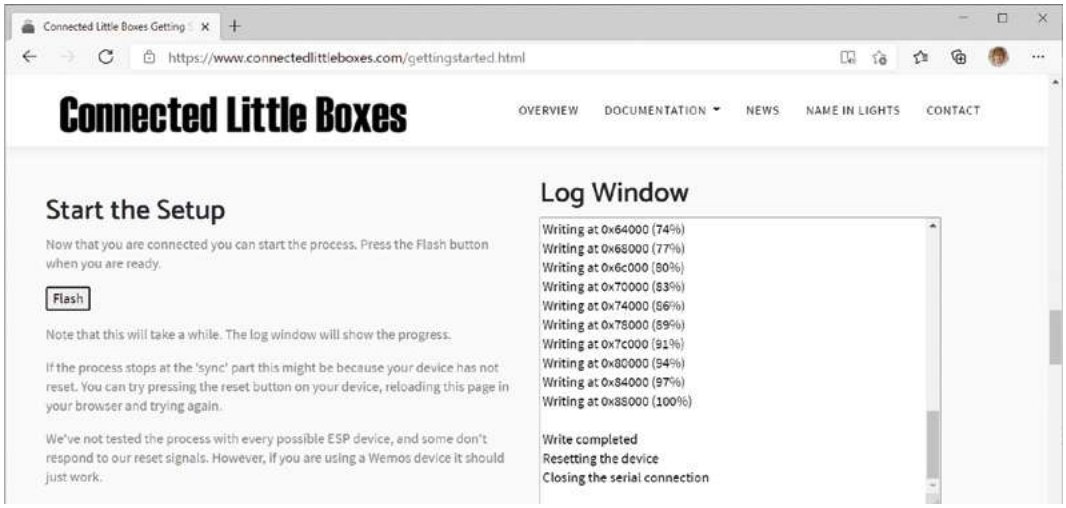
This command might look a bit hard to understand, but in English it means "Tell the console process to send a JSON report triggered by the BME280 temperature sensor each second. Put this command in the start store so that each time the device starts the command is performed". We just have to enter the command once to start the sensor running, and it will perform the command each time it is powered on. You can use Simpleterm to connect to a box via a browser and send this command into it.

Of course, before we can talk to the Connected Little Boxes software, we need to load it into the



Above
The shadow effect is achieved using a custom style sheet, which you can find on the website

Right
A Connected Little Box environmental sensor



hardware. We could compile the source code of the software on our computer and then download it into the device from our computer, but it turns out that there is a much easier way of doing this.

FLASHING FROM THE BROWSER

When you deploy a program from the Arduino development environment, the program is sent via the serial port into a 'bootloader' program in the device that burns the code into the device. This process is called 'flashing' the device, since the code is written into Electrically Erasable Programmable Read-Only Memory (EEPROM), which is also called 'flash' memory.

However, now that a browser can use the serial port, it is possible for the flashing process to be performed directly from a web page. This means that you can buy a new device, plug it into your computer, go to **connectedlittleboxes.com** to flash the device, and then use Simpleterm to set it up.

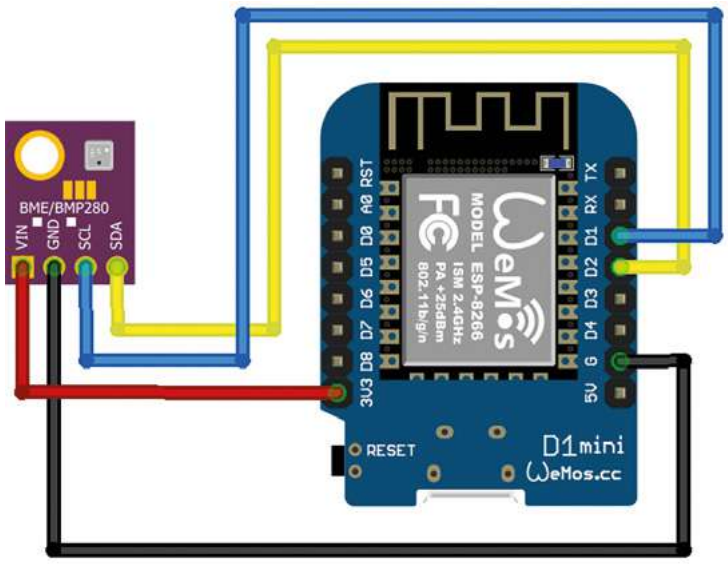
Using the browser to talk to devices over the serial port could make a huge difference to the way that we program and use embedded devices. We no longer have to load any programs onto our machine – we just need to visit a web page and can do everything from there.

WHAT DOES AWAIT MEAN?

Wouldn't it be nice to be able to make another instance of yourself whenever you need to do something boring? Rather than standing in line at the supermarket checkout, you could just leave your duplicate waiting while you go home to do something more interesting. Then at some point in the future, your duplicate will turn up with the shopping.

This is what the `await` keyword does in the Simpleterm program. Some actions involved in using serial connection take a while to complete. These are marked with the `await` keyword. When the program reaches an `await`, it creates a new execution path to perform the awaited action. The program then exits the method before the awaited action is complete.

Functions containing awaits must be marked as 'asynchronous' because the actions they perform are not synchronised with the return from the function – they complete (or fail) at some point in the future. Asynchronous functions are a great way to keep a website responsive, so that a JavaScript function running in response to a button press doesn't stop other elements on the page from responding to user actions.



Above
Flashing software from the browser

Below
These are the sensor connections. Note that the BME280 must be powered from the 3.3V power source on the ESP8266

HackSpace
TECHNOLOGY IN YOUR HANDS

FIELD TEST

HACK | MAKE | BUILD | CREATE

Hacker gear poked, prodded, taken apart, and investigated

PG 110

DIRECT FROM SHENZHEN:
BLUETOOTH AMP
Add wireless connectivity to your speakers

PG 112

PICO ROBOTICS
Add motors and servos to your microcontroller

PG 104

BEST OF BREED
Our pick of the best accessories for Raspberry Pi Pico

ONLY THE BEST

Friends for your new Raspberry Pi Pico

A few handy Raspberry Pi Pico accessories

By **Marc de Vinck** @devinck

When most people think of Raspberry Pi, they think of a dessert. But this probably isn't a mistake made by anyone reading this article. We all think of the amazingly affordable, single-board computer. The important part to note is that it's a 'single-board computer', not a microcontroller. Well, that thinking is old-school now thanks to the Raspberry Pi Pico, the latest product to join the range. This new product is an incredible little microcontroller that we're sure will find its way into many projects over the next few years.

But what makes the Pico so special compared to other microcontrollers? In many ways, it's quite similar to other microcontrollers on the market , and that's a good thing! Pico can take advantage of established ecosystems such as CircuitPython and Arduino (as well as MicroPython and it's own C/C++ SDK). Despite this, Pico does have a few tricks up its sleeve that make it really quite special. Perhaps the biggest feature is the price. The development board costs just $4! Of course, it's not just about price, and Pico has a few features that make it stand out. These come from the RP2040 chip that sits at its heart.

The RP2040 features a dual-core Arm Cortex M0+ processor, and has 264kB internal RAM and support for up to 16MB of off-chip flash. You also get a slew of flexible pins that support I2C, SPI, and Programmable I/O (PIO for short). It makes the Pico a great choice for beginners, as it shares a lot of common features that are well-documented,

> This new product is an incredible little microcontroller **that we're sure will find its way into many projects**

and for professionals because of all the flexibility, customisability, and advanced features. I picked up a handful when they first came out, and I've been really impressed so far.

Although the Raspberry Pi Pico hasn't been around for a long time, there are already a decent number of Pico-specific accessories available. And I'm sure there will be a lot more released every week. Here are some of our favourites that are available right now, but keep an eye out for new ones!

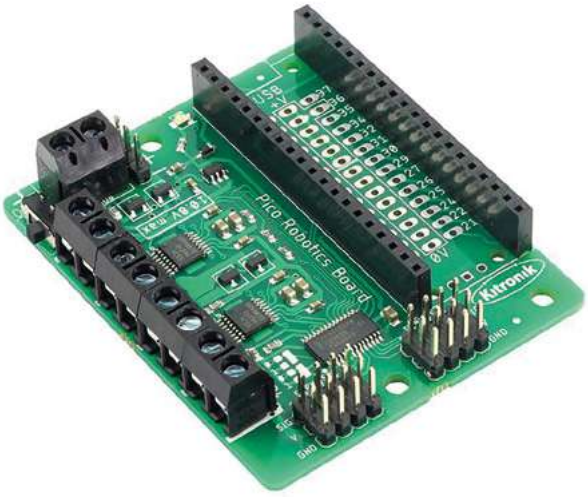
Kitronik Robotics Board for Raspberry Pi Pico vs Pico Explorer Base

KITRONIK £13.20 | kitronik.co.uk

PIMORONI £22.20 | pimoroni.com

Building a robot with your Pico? Then you should be looking into the Kitronik Compact Robotics Board for Raspberry Pi Pico. Simply connect your Raspberry Pi Pico to the breakout board via the female pin headers, and you immediately get the ability to control four motors, or two stepper motors, and eight servos.

The board adds all this functionality by adding two dual H-bridge motor driver ICs and the necessary components, like power supplies and additional ICs that make building a bot with the Pico simple. It does add a bit of bulk to a project that's built around the small form factor of the Pico, but it's worth it as you'll save so much time when using this handy breakout board to control multiple motors. Just plug and play! No soldering required.

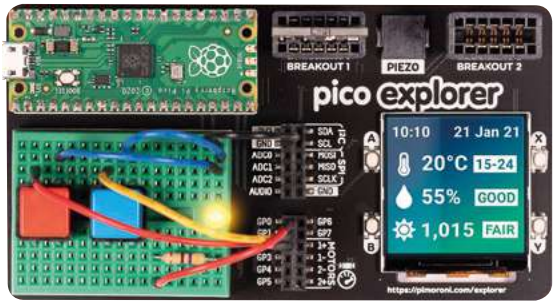


Left
Lots of high-power outputs

Below
A lot to experiment with on one board

The developers over at Pimoroni say their Pico Explorer Base will transform your learning into an "electronic adventure playground", and I happen to agree. Inevitably you'll want to add a motor or screen to your Pico at some point, and if you do, then this breakout board is for you! But that's just a little of the added functionality that this board offers.

In addition to the 1.54" IPS LCD screen, and two half-bridge motor drivers, you'll also find a piezo speaker, four user-controllable switches, two Breakout Garden I2C sockets, a mini breadboard, and convenient access to the GPIO and ADC pin headers. Oh, and some rubber feet to add to the back, which is often overlooked, and important for reducing electrical shorts when prototyping on your bench.



The Pico Explorer allows you to experiment faster, quicker, and easier. If you want to test a few theories on your Pico, or build a bot, hook up some sensors, or just have fun, then this is a good tool to pick up along with the Pico. →

VERDICT

Kitronik Robotics Board for Raspberry Pi Pico

A very handy board for bot-makers.

9/10

Pico Explorer Base

A great choice for experimenting.

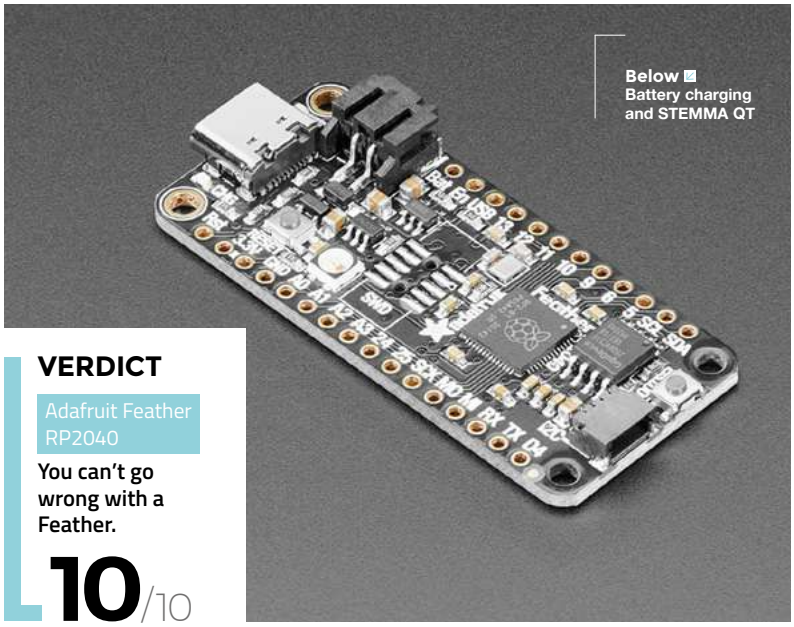
9/10

Adafruit Feather RP2040

ADAFRUIT $11.95 | adafruit.com

Not everything in this Best of Breed roundup is an accessory for the Pico. The Adafruit Feather RP2040 joins the extended family of Raspberry Pi products by integrating the custom-designed RP2040 microcontroller IC from Raspberry Pi into Adafruit's incredibly popular Feather line of products. And if you've never used a Feather board from Adafruit, you're in for a treat! It has a perfect form factor and has lots of compatible sensors and accessories, along with a robust community of developers.

The popular Feather family now includes the RP2040 32-bit Cortex M0+ dual-core running at ~125MHz, 8MB SPI flash, 21 × GPIO pins, built-in 200 mA plus LiPo charger, an RGB NeoPixel, STEMMA QT connector, and more! Head over to the product page to get all the details about the latest addition to the Feather family. While you're there, check out all the other Feather microcontrollers and add-ons. You won't be disappointed!



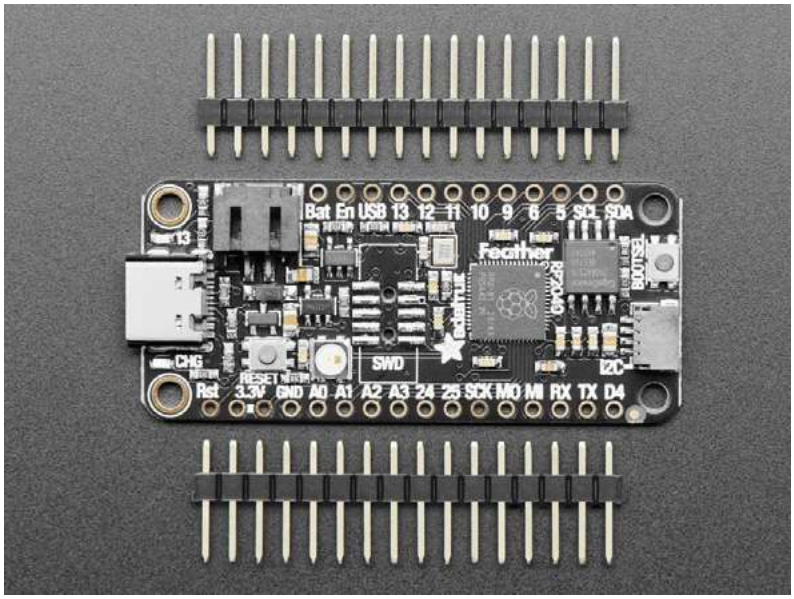
Below
Battery charging and STEMMA QT

VERDICT

Adafruit Feather RP2040

You can't go wrong with a Feather.

10/10

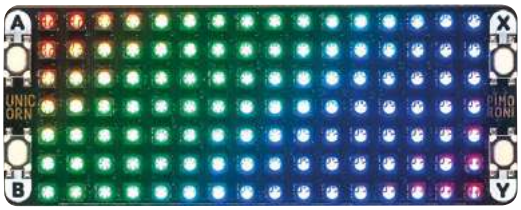


Pico Unicorn Pack

PIMORONI £19.80 | pimoroni.com

Left
So. Many. LEDs

Who doesn't love LEDs? Nobody! And the Pico Unicorn Pack from Pimoroni satisfies all our inner LED needs in a tiny RGB form factor. They packed 112 RGB LEDs into the same footprint as the Pico, and they were even able to sneak in four buttons for some much-needed additional functionality. Lighting up LEDs is fun, but you will inevitably want to add some interactivity, and those additional buttons are a nice touch.



The board fits perfectly on top of the Pico. You can control the colour and the brightness of each LED individually. The board comes pre-soldered with all the LEDs and header pins, so you'll be up and running quickly. Head to the website to learn more about how to program the Pico Unicorn Pack, and find links to Python libraries and examples. →

VERDICT

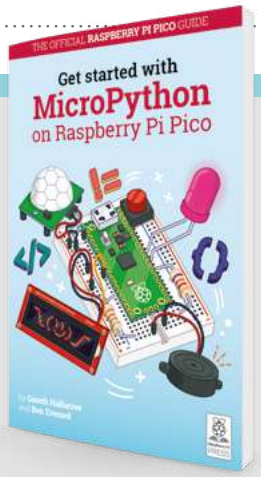
Pico Unicorn Pack

Love LEDs? This is for you!

10/10

GET STARTED WITH MICROPYTHON ON RASPBERRY PI PICO – 2ND IMPRESSION

RASPBERRY PI £10 | store.rpipress.cc



Left
A little bedtime reading

Sometimes a physical book is nice to have when learning electronics. Yes, you can find a lot of information online, and even a PDF of this book, but if you are like me, sometimes you just want a printed book. *Get Started with MicroPython on Raspberry Pi Pico* is the perfect companion for anyone learning about microcontrollers, especially the Raspberry Pi Pico. It's the official guide from Raspberry Pi Press, so you know it's great, and filled with example projects.

BEST OF BREED

Pico RGB Keypad Base

PIMORONI £21.90 | pimoroni.com

LEDs are fun, but LED buttons are more fun! And soft, silicone buttons are the most fun! New from Pimoroni, known for its beautiful LED boards, is the Pico RGB Keypad Base for your Pico. The breakout board features a 4×4 matrix of soft, rainbow-illuminated keys that are perfect for creating things like a music controller, interactive visualisers, or even creating a simple game.

When you connect your Pico to the board, via two rows of female header pins, you'll instantly have access to the LED Keypad matrix via I2C. There are also conveniently labelled breakouts for all the pins, so you can easily solder additional wires and sensors as needed.

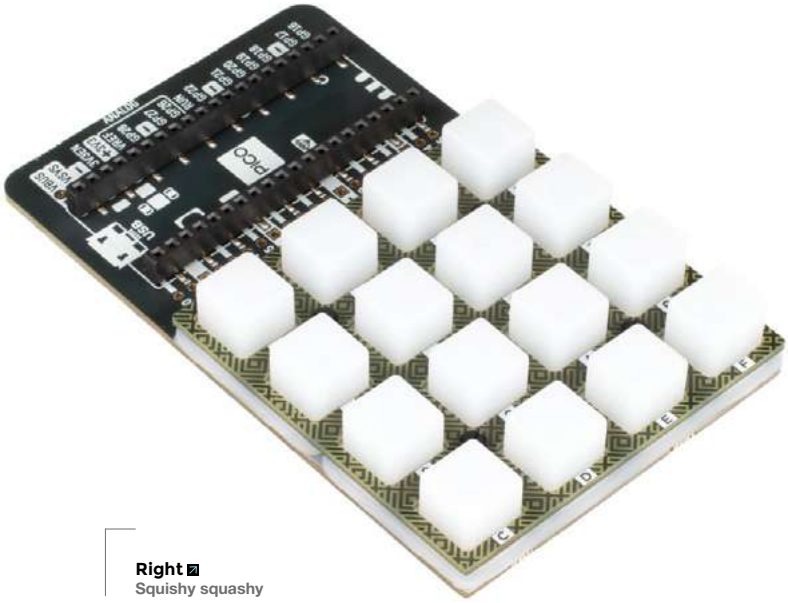
The website has more information about the Pico RGB Keypad Base, and lots of code examples written in Python that you can download and test. □

VERDICT

Pico RGB Keypad Base

LEDs and buttons! What's not to love?

9/10

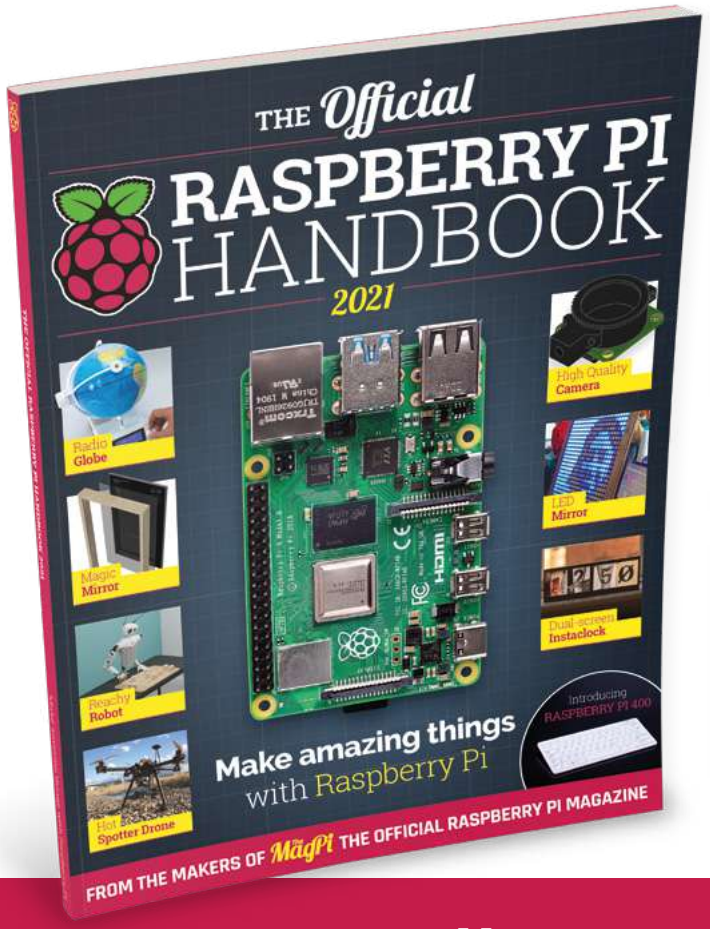


Right Squishy squashy

THE *Official* RASPBERRY PI HANDBOOK 2021

200 PAGES OF RASPBERRY PI

- Get started with Raspberry Pi, electronics, and more
- Be inspired by incredible projects made by other people
- Learn how to code and make with our step-by-step tutorials
- Find out about the top kits and accessories for your projects



Buy online: magpi.cc/store

AK370 Bluetooth stereo amp

Listen to your favourite tunes using this cost-effective amplifier

By Jo Hinchliffe @concreted0g

Available for between £12 and £21, the AK370 amp is quite a lot of technology stuck into a pretty rugged metal enclosure. It arrived well-packed, with just the amp and a small, plastic remote control. The amp doesn't come with a power supply (and indeed the remote doesn't come equipped with a battery), but it takes a 12V supply via a small, common barrel jack, so we didn't have to look far to find a suitable supply.

Below
The front panel of the AK370 Bluetooth amp. Note that the LED display works fine, but is hard to photograph!

It suggests on the packaging that it isn't fussy about what type/impedance speaker drivers you attach to it and, to test, we had a set of small 4Ω speakers in our parts hoard. Speaker connections are made with small, sprung clip connectors that we hadn't seen the like of for a while, but used to be the most common speaker connector on mass-produced domestic hi-fi systems. Also in the box is a small connector and half a metre of wire that can be plugged in at the rear as an antenna for the FM radio.

Powering on the unit, we found it paired with our Android phone with no problems at all and works perfectly as a phone speaker for listening to music, podcasts, or for watching video content. Whilst this is definitely not destined for use by audiophiles in high-end systems, it's true to say that this thing is loud and also clear, with no audible distortion. It was certainly powerful enough to drive our admittedly small speakers too hard, and would probably sound even better with larger drivers.

There's volume and bass and treble adjustment on hardware ports on the unit, and a small selection of buttons that allow you to switch between modes, Bluetooth, line input, USB/SD card, and FM radio.

Without using the remote, it doesn't seem possible on our device to tune the FM radio, but once we popped a CR3032 battery into the remote, it was pretty easy to do so. However, whilst we've worked out how to tune the radio and save station presets, we haven't worked out how to switch between them on

Left
Rear panel of the AK370 showing the speaker, power, and line-in connectors

Below
Some functions of the AK370 are only available on either the hardware unit's buttons or the remote control

the remote! We loaded a USB drive with a few albums of MP3s, all arranged into album folders placed in the root directory of either the USB drive or the SD card. It worked flawlessly to find the MP3s and play them, and we could quickly skip through tracks. It also (unlike other cheap MP3 circuit boards we have used previously) manages to ignore album art image files and bonus files in album folders, which is excellent. There's a random setting accessed via the remote, that allows playback to be limited to the end of the current track, rather than continual play, and we were pleased to note that even after a power-down, if the USB or SD card hasn't been removed, it will continue playback from the track position when the unit is next powered on. That's pretty crucial if you don't want to have to scroll through lots of tracks each time you use the device.

The line input is via two phono sockets on the back of the unit and, again switching to line input, everything works well. Handy, we guess, if you wanted to use the amp for playback from something without Bluetooth, like a portable digital recorder.

We aren't quite sure what the intended use case for this amp is – being 12V, it's useful in vehicles, and we can imagine it being used as a device perhaps in a camper conversion. Likewise, it's cheap and rugged enough to be used in a workshop or shed, it takes up little room, and could be screwed/bolted to a shelf or the underside of a workbench with ease. It runs consistently, seems reliable and, as a budget device, we did think it could be very useful for art installation-type work.

As for ours, we think it's destined to be mounted into a scrap-built box containing a couple of speaker drivers – a decent shed solution that can run off an old 12V bench supply, or a 12V battery if needed. ◻

> **Powering on the unit, we found it paired with our** Android phone with no problems at all

DIRECT FROM **SHENZHEN**

REVIEW

Kitronik Robotics Board for Pico

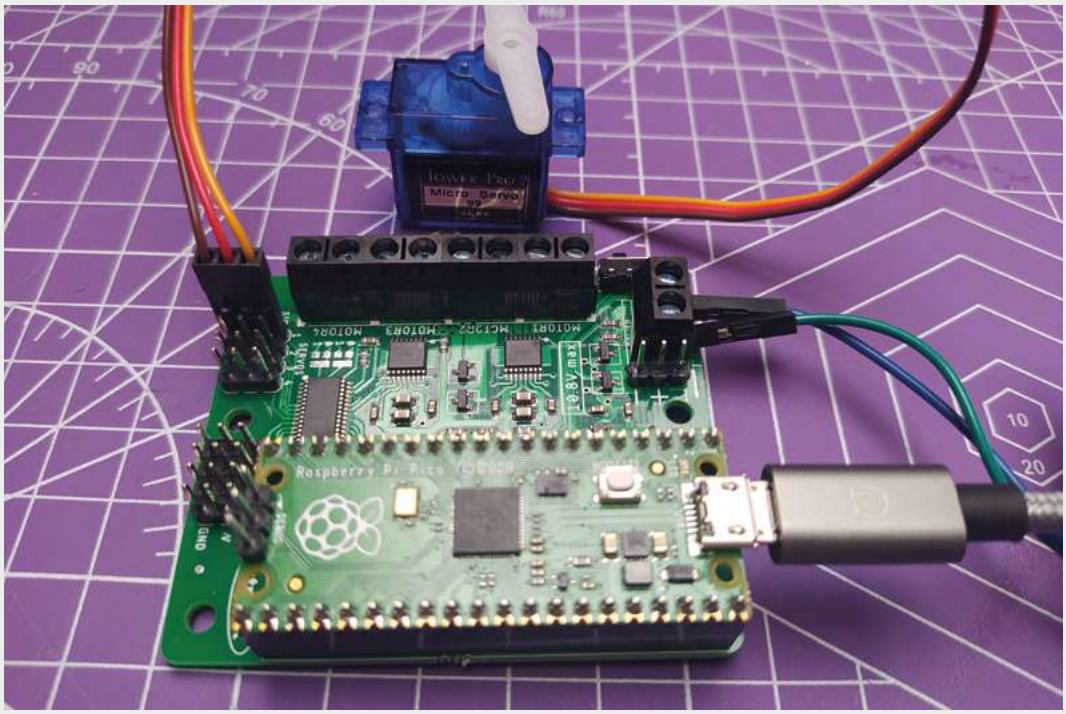
Make your microcontroller move

KITRONIK £13.20 | kitronik.co.uk

By Jo Hinchliffe @concreted0g

The Pico robotics board is a new device from Kitronik aimed at getting you up and running with robotics projects on the Raspberry Pi Pico as quickly as possible. It has a huge range of features including the ability to drive up to four DC motors bidirectionally, driving up to eight servos, and the ability to drive stepper motors. On paper, the Kitronic Robotics Board for Pico is a very capable board, but does it live up to this?

With everything soldered and pre-installed, your Pico slots into a set of headers. The USB port is still available for programming the Pico, but the board itself has its own power input and a hardware on/off switch. The addition of a power switch is great as it can potentially act as an on/off switch for your entire robot project. You can wire up a power supply in the range of 3.0V to 10.8V, but the suggested useful range is 6–10.8V. Power connections are made via a terminal block; alternatively, there is a servo-style connector



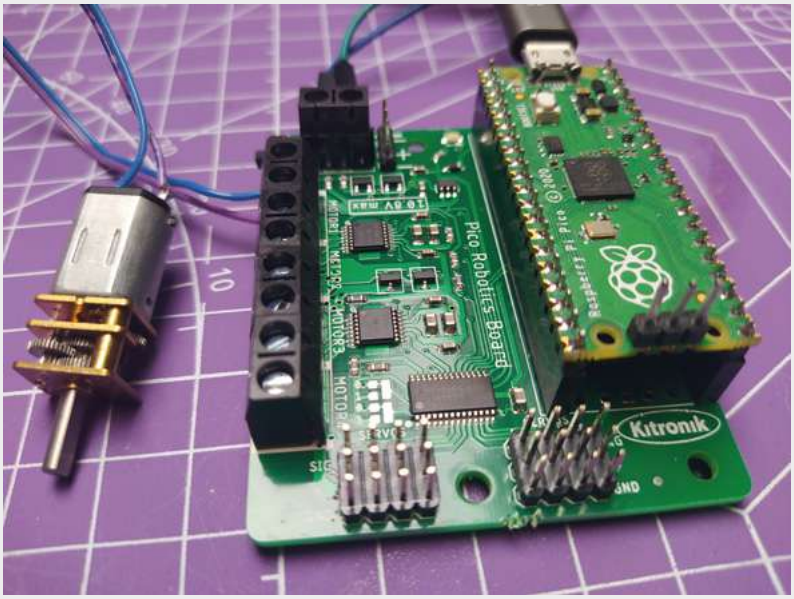
Right
Testing a servo with the board. As it can drive up to eight servos, this board is a good target for complex movement or multi-jointed arm projects

available as well. This is a great design choice, as many people building robots may be using hobby RC-type components, such as battery eliminator circuits (BECs), so servo connectors may be readily available.

Connections for the DC motors are again made with terminal blocks and screws, and the connectors for servos are the standard three-pin servo headers. We are pleased to see that this robotics-aimed board has bidirectional control of four DC motors. There are a lot of boards on the market that only seem to offer bidirectional control of two motor channels, which is fine for smaller robots or rovers with tank-style steering, but if you want to build a true AWD or a Mecanum-wheeled robot, then having four bidirectional channels is extremely useful. There's a maximum continuous current rating of 12A for all attached motors and servos, and each motor channel can handle up to 1.5A. This amount of power opens up a good range of DC motors that you can run with this board.

Your Pico mounts to the board using header pins, and there's a big USB marker on the Kitronik board to ensure you orient the Pico correctly. The Pico sits over an area that has three rows of copper pads, with the outer rows breaking out the remaining 27 I/O pins that the board isn't using. The centre rail has positive and negative pads to connect additional items to the robotics board power supply. The power supply for the board and motors is regulated and used to power the Pico. Also of note is that if you plug in the Pico USB to power and program the Pico, the USB power is not passed through to the board. It means you can power down your robot experiment and hook up the USB to tinker with your code without worrying your robot will suddenly make a leap for the edge of your desk.

> "There's a maximum continuous current rating **of 12A for all attached motors and servos**"



Above
It's simple and straightforward to get up and running with the code examples

Below
The board is well-thought-out, with numerous mount points and other practical features, such as the on/off switch

Links are supplied to the GitHub repository for a MicroPython module and some example code. There's also a handy Kitronik blog post linked on the product page and on the GitHub repo, talking about how to get the examples up and running. It's pretty straightforward – download the zipped file from GitHub, extract the contents, and then copy the **PicoRobotics.py** file onto the Pico. You can then explore and run the other example codes supplied. Using a bench supply set to 7.4V (emulating a two-cell battery we often use in robot builds), we worked through a couple of tests. We connected a DC motor and ran the **SingleMotorTest.py** code which turns on each motor channel in turn, ramping the speed up and down in one direction and then the reverse direction. We repeated this process running the **SingleServoTest.py** code with a 9g micro-servo attached. Both worked perfectly and the code is commented well. The README file on the GitHub is also a good primer on writing simple scripts for this board. One thing to watch out for is that if you save the examples to the Pico using Thonny and then run the script from Thonny without the robotics board attached, you will get error messages as it can't detect the board.

The Pico Robotics Board has a nice size and layout, with multiple M3 mounting holes. It's small enough that it would fit into a Mini Sumo class robot chassis, but its feature set and driving capabilities mean it could be used in much larger and more complex robotics projects. ◻

VERDICT
An excellent, well-thought-out robotics board with a heap of capabilities and great example code.

9/10


"I don't want to just play a role; I want to be genuinely excited about the things I do."
Simone GIERTZ